COMPUTER RECOGNITION
OF
PARTIALLY-OCCLUDED OBJECTS

by

Chan Ming-Hong

(陳銘康)

A MASTER THESIS SUBMITTED IN PARTIAL
FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY
IN
THE DEPARTMENT OF ELECTRONICS
THE CHINESE UNIVERSITY OF HONG KONG

HONG KONG

JUNE, 1986

Thesis
TA
1632
C43

471314

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude towards my supervisor, Dr. H.T. Tsui for his kind guidance and encouragement throughout the course of this work.

Thanks are also due to Mr. M.F. Mak for his help in printing this manuscript, and to the Croucher Foundation for the financial support they provided in the past two years.

ABSTRACT

Recognition of partially-occluded objects is an essential task for many industrial applications of machine vision. A fast and reliable method for recognizing partially-occluded, arbitrarily-shaped objects for both two and three dimensions is proposed. There are two principal components to the method. The first is a subtemplate matching scheme that breaks the object to be recognized down into smaller parts called subtemplates for matching. For the 3-D case a new and efficient algorithm that works on depth map representation of the object is proposed. The algorithm attempts to estimate the orientation of a surface patch and perform the matching using mainly the outer boundary of the patch. The second main component of the method is a constraint application process that selects among a number of probable matches that result from the subtemplate matching stage an optimal consistent set to form a solution. The process is formulated as a Dynamic Programming problem. Experimental results show that the algorithm works well on objects that are truly arbitrarily- shaped, with a great reduction in computing time compared to previous schemes.

CONTENTS

CHAPTER 1

INTRODUCTION

With the advent of digital computers there has been a constant effort to expand the domain of computer applications. One area of interest that much of the effort has been spent is that area of artificial intelligence that we shall refer to as machine perception.

Among the various applications of computers, perception is perhaps one of the most difficult problem yet to be solved. It is a well known fact that a 5 year old boy probably has difficulty in solving a system of linear equations whereas he can give a rather precise and detailed description of a scene presented in front of him. He will be able to tell you that there is a white house with a tall tree next to it with a swimming pool in the front where three children, two boys and a girl are playing. The reverse is true for a computer. The difficulty lies in part in the fact that we know little about human's cognitive process. We are practically all experts in perception, but the knowledge of the process itself is too little to do much help in designing a machine to resemble human's ability in perception. The difficulty lies not only in the limited ability of the machines to perceive their environment (as a matter of fact many different types of transducers, such as light, sound, temperature, range finders, pressure, tactile, infrared and etc. are available), but on the interpretation of the perceived data. We are in the most general

sense interested in answering such questions as "Describe the scene." Motivation for answering such a question may arise from some practical applications. For example the versatility and power of a moving robot are greatly enhanced when they are provided with visual abilities. The problem associated with the attempt to answer such questions is similar and in fact related to the general problem in most knowledge-based system. The saying that "intelligence requires knowledge" in certain respect reflects where the problem as well as the solution lies. To give a general and detailed description of a scene requires a substantial knowledge base. Depending on the details we want the size of the knowledge base may grow to a point that the required time to search for a solution is too long to be practical unless some very clever mechanisms for structuring and maintaining the knowledge base are devised. At the meantime it is worthwhile to say that it is still a long way to get a machine with perception capability comparable to a human being. The progress is only possible with the advent of artificial intelligence.

In spite of the difficulty, machine perception does have great success, or at least solid achievement in specific problem domains, just as we have very successful expert systems as products of AI research. One such example is character recognition which is in some way a rather simple but useful task. Although one easily realise the difficulty when the characters are made not just by machines with different fonts but may be made by hand with different authors encountered, there exists many successful techniques as long as the situation is clearly specified and

under good control. The techniques are every now and then being refined, and new approaches emerges which can deal with more and more complex situations.

In applications where the task is to find out whether a known entity, presumably stored in the computer's memory with others to form a relatively small model set, is present in the scene, or to classify an unknown object in the scene as belonging to one of several classes of known objects, the problem reduces to one so called pattern recognition and classification. Character recognition, as mentioned above, is one such example. Others include automatic inspection of machine parts, aircraft identification, finger-print analysis, identification of ground targets from an aerial imagery, recognition of tumors in chest radiographs and human face identification, just to name a few. As a matter of fact most researchers nowadays are more or less directed towards finding solutions for spacific problem domains, rather than devising general schemes to encompass all problems at hand, since one can never expect a single technique or a small number of different techniques can be applicable to such a variety of problems.

In solving the above so called pattern recognition and classification problem which can be viewed as the preliminary process in obtaining a general relational description of a scene, one typically has to train the machine in the first hand before actual recognition is done. In the training process an object or a class of objects to be identified is abstracted by certain features to form a feature vector, which together with feature

vectors of other objects or other classes constitute a feature space to be stored in a library. During the recognition or classification process the suspected object goes through the same feature extraction process. The extracted feature vector is compared with those from the library, and the suspected object is said to belong to a certain class or to be a certain object if the distance between their corresponding feature vectors is minimum and also below a certain threshold. In such a scheme one typically has to deal with i) feature design, ii) feature extraction, iii) searching a feature space and comparing feature vectors. The above scheme can be better illustrated with the diagram shown in fig.1.1.

One important issue in the above scheme is feature design. A feature can be a relatively low level representation of the object under study, for example a list of x-y coordinates representing the outer boundary of a two dimensional object, or it can be a very abstracted representation such as an outcome of applying the symmetric axis transform [1] on the silhouette of an object. A low-level feature usually results in a relatively simple feature extractor, whereas for high level features the extractor usually turns out to be more complicated, but the resulting library usually takes less space since the representation is more abstracted, also the matching of feature vectors turns out to be more simple and robust.

The research work presented in this thesis is motivated by the attempt to automize the inspection process in a typical

industrial manufacturing line. As a basic step for automatic inspection, the machine parts or in general the objects to be inspected must be properly positioned and oriented before actual inspection can be done. In this research a new and efficient scheme was devised to position and recognize both two and three dimensional objects with capability to handle poor lighting condition and partial occlusion. It is well known that machines with such capabilities are much more flexible and useful compared to those that work only on isolated objects under perfect illumination. As a matter of fact with such capabilities the scheme applies well on many other situations.

When dealing with the occlusion problem, most schemes that based on matching the entire object as one single entity fails since the occluded object in the scene to be recognized no longer resembles the entire unoccluded model except for some visible parts. One natural approach to handle this is to break the object down into several homogeneous sections instead of treating the entire object as one single entity, hoping that those sections that are not occluded can be useful in inferring the solution. In fact this is the basis for most techniques that deals with occlusion. However when these smaller sections are considered as individuals during the recognition process, the individual results from matching each of them against the apparent scene may turn out to be inconsistent. Each section may find itself matching the object at more than one instance and obviously some of them are false matches. Due to noise and other factors the best match may not necessarily be the correct one, and this is

where the ambiguity comes from. Therefore a question arises as to how the individual results can be put together to infer a consistent and meaningful solution with respect to some domain-specific constraints such as continuity, closure and rigidity. Therefore one has to add another stage in the process that pieces up the individual results to infer the solution. This is depicted in fig.1.2.

One potential advantage of such a scheme that one may probably miss, as compared to that of fig.1.1 even when no occlusion problem exists is that the matching of individual sections can be done in a parallel fashion. This is especially true when low level features are used since the matching takes the largest time slice in the entire recognition process, while on the contrary the final stage in fig.1.2 can be a quite efficient process, especially if no occlusion exists. The gain comes from the fact that matching an entire object takes longer than matching just a fragment of the object. As long as the matching of fragments is done in a parallel fashion, we can remove the bottleneck of the entire process.

The approach we took in this research is to use low level features, to perform matching in fragments, and to apply rigidity constraints to infer the solution. The process to match in fragments is usually referred as sub-template matching. Since low level features are used, the feature extraction process becomes relatively trivial. Therefore the algorithm proposed in this work is mainly a two-stage algorithm where the first stage is typical subtemplate matching. For the 2-D case a well-known matching-in-

θ-s-space method is used in matching the subtemplate with boundaries of the object from the scene. For the 3-D case a new algorithm that works on depth map representation of the object is proposed. The algorithm attempts to estimate the orientation of a surface patch and perform the matching using mainly the outer boundary of the patch only. The outcome of the first stage can be viewed as some partial results. In order to piece them up one of several methodologies can be adopted. The simplest one is to accumulate the partial results in a statistical manner to get a solution. Others achieve the same goal by exploring every possible combination and select the best with respect to some constraints. Another common technique is stochastic labeling which explicitly attempts to optimize a criterion function. Such a constraint application process is formulated as a dynamic programming (DP) problem in this work.

This thesis is divided into two main parts. Part I deals with the two-dimensional case whereas Part II deals with the three-dimensional case as a generalization. Although the basic scheme remains the same for both cases, namely subtemplate matching then constraint application, each of the two cases has its own emphasis. Subtemplate matching in two dimensional is an old subject, but the way the partial results are pieced up to obtain an optimal solution using DP is new. Low-level subtemplate matching in 3-D, on the contrary, is seldom attempted. Therefore more emphasis is put on this subject in Part II.

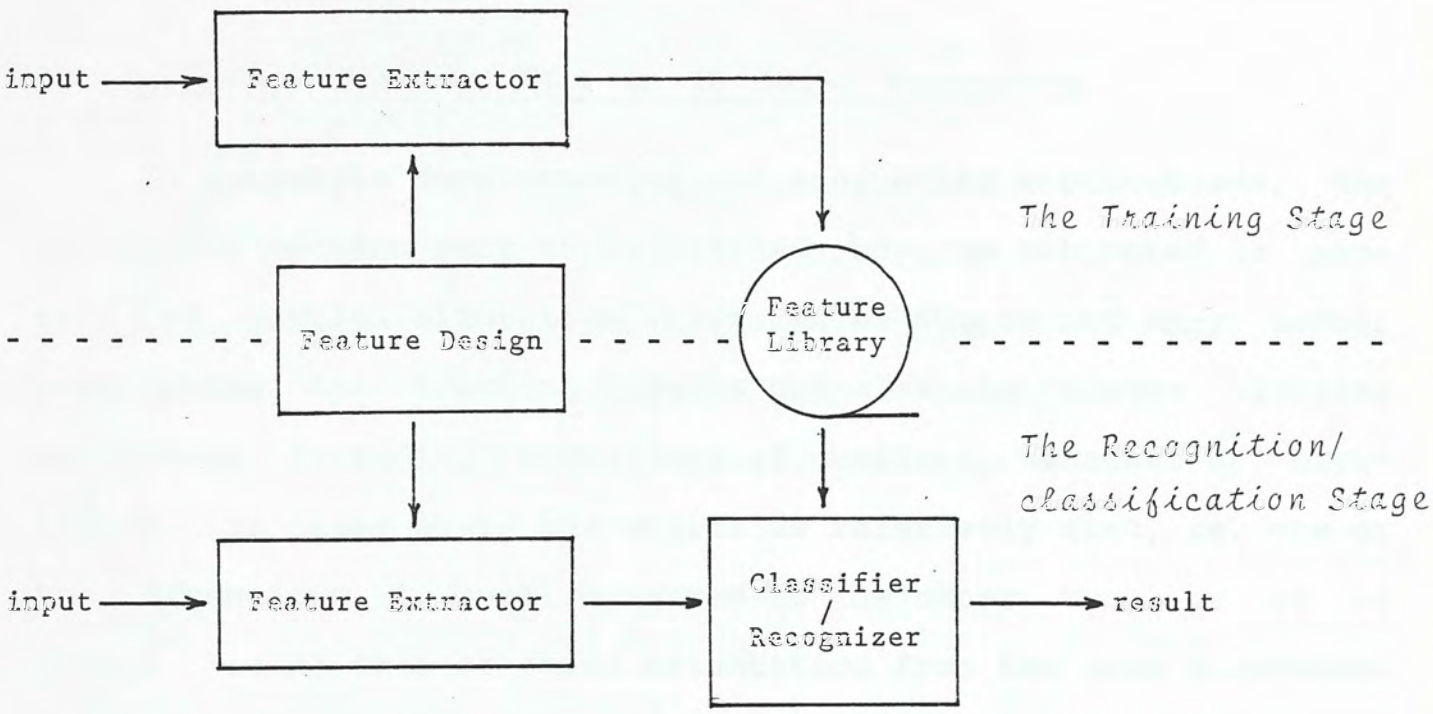


Fig 1.1 The general pattern recognition/classification scheme

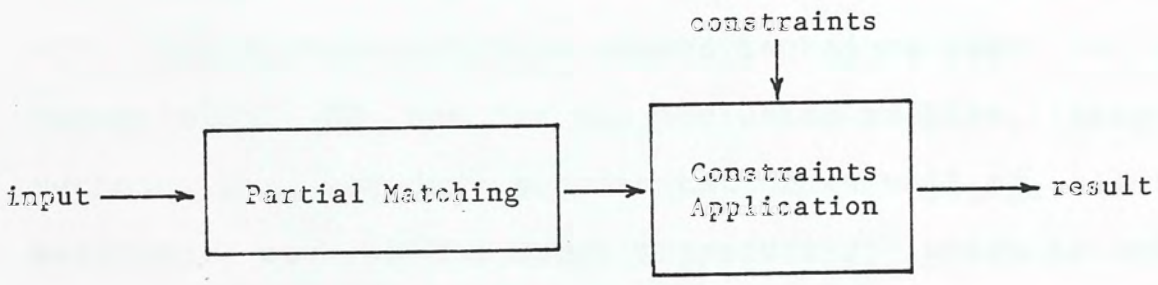


Fig 1.2 Recognition based on partial matching & constraint application

CHAPTER 2

RECOGNITION OF PARTIALLY-OCCLUDED 2-D ARBITRARILY-SHAPED OBJECTS

2.1 Review of Previous Work in 2D Shape Recognition

In automatic manufacturing and many other applications, the object or machine part to be fetched may be subjected to some sort of partial alteration which can be due to not only actual overlapping and touching objects but also to uneven lighting conditions including variations of contrast, shadows or highlights. In cases where the object is relatively flat, ie. one of the dimensions is small compared to the other two, or it is always 'seen' at a constant orientation from the same direction, the boundary or silhouette of the object is an adequate pattern for representation and matching. The computation of image boundaries is relatively easy and can be done at a high speed by special purpose hardware. However, the image boundaries are in general resulted from multiple objects occluding one another. Recognition by simple template matching will not work.

Shape recognition based on partial match of model boundary with image boundary is a common technique used in 2-D object recognition. If not for the occlusion problem, many efficient methods for boundary representation as well as algorithms for matching, such as the Hough Transform[2], which is equivalent to template matching, the Fourier Descriptor technique[3] and the moment invariant method[4] would have been effective. Template matching and its equivalents, as demonstrated by Sklansky[5], can

be viewed as performing the function of a matched filter on the incoming pattern. However it is no longer optimal in recognizing one shape in the presence of others. Fourier Descriptors and moment invariant methods suffer from the same problem in that the transformed representation of an isolated object bears no simple relationship on that of the apparent object formed with the presence of others. However they do have their applications where the shapes to be recognized are in most cases isolated, such as hand-printed character recognition and aircraft identification where the craft forms a clear silhouette against the background sky.

One major approach to handle the occlusion problem is the segmentation-labeling method[6]. The object boundaries are first segmented, usually into straight lines and circular arcs. During the learning stage these segments are stored in the library and form the feature space for matching. When actual recognition is performed, these segments are compared with those from the library and a correspondence is made between the two sets of segments, usually by means of graph matching or relaxation techniques. McKee and associates[7] used nodes as the primitive elements for matching. Perkins[8] used 'concurves' to represent and match the objects. Bhanu[9] used straight line segments and applied stochastic labeling to match the segments. Since the outcome of segmentation is sensitive to many factors(two similar-looking objects after segmentation may turn out to look very different for an human observer), sophisticated relaxation or graph matching has to be applied to get an unambiguous and

consistent solution. The computation associated would be quite labourious. In those schemes, the matching of features are rather trivial since high level features are used. The work load is transfer schemes, the matching of features are rather trivial since high level features are used. The work load is transferred partly to the feature extraction process, namely segmentation, and partly to the complicated scheme of constraint application such as relaxation. The performance of such schemes deteriorate especially when highly irregular shapes are concerned, since the segmented output which is an approximation of the shape itself either turns out to be inadequate or too complicated for a relaxation scheme. However, these approaches have the advantage that they can effectively handle size variations as well as visually noisy images.

Another major approach towards the occlusion problem is subtemplate matching done in a low-level pixel by pixel manner. The basic idea is to break the template down into many smaller subtemplates and perform template matching with each of them. There exists techniques similar in nature to Hough Transform to detect arbitrary shapes[10]. Turney and associates[11] generalized the concept and used subtemplate matching for recognition in a statistical manner. In their scheme, each model boundary is broken down into many overlapping subtemplates, each of which is matched against the image boundaries. The matching results are 'accumulated' in an accumulator array which covers the entire image to be analysed. No segmentation except boundary extraction is required. Since no contextual information is used, no relaxa-

tion scheme nor symbolic matching of any sort is exercised. They also introduced the concept of 'saliency' which measured the relative importance of each subtemplate and used it as a weighting factor when incrementing the accumulator array. Their method is computationally expensive as a large number of low-level subtemplate matchings are required. Variation in scale is not allowed as it will add another dimension to the match space. The scheme is simple and attractive for detecting highly irregular shapes, especially those objects that are very similar except for a few distinquished portions. Matching in low level has the advantage that fine details can be handled which would have otherwise been missed by the first approach, but this is also a curse since this implies that the scheme is more sensitive to noise. However since each subtemplate is treated individually, the corruption of one or a few by noise usually does not render the scheme from achieving a correct solution. Nonetheless when such a large number of subtemplates are used in a statistical manner as Jerry et.al. did, the time required to compute a solution, which is mostly spent in performing low-level matching, is usually too long to be practical for most industrial applications unless they are performed in a parallel fashion. In this work, we propose an algorithm based on subtemplate matching, but will compute a solution much faster than other known subtemplate schemes.

We present here a two stage algorithm which incorporates a scheme of using contextual information explicitly. In the first stage we perform subtemplate matching. Although our algorithm can

work with overlapping subtemplates, non-overlapping subtemplates should be used whenever practicable. This reduces the number of subtemplates by a factor of ℓ compared with the full overlapping case, ℓ being the length in pixels of the subtemplates. For example, if ℓ equals 20 and the boundary length is 100 we match only 5 subtemplates instead of 100 for the full overlapping case. At first glance this might seem crude. A small number of subtemplates will not work using the usual match-accumulate scheme, but will give reasonable results by our algorithm. Instead of using an accumulator array to store the matching result, we explicitly set up a match table which stores up to m best matches for each of the subtemplates used. This limits the size of the set of candidate matches and greatly reduces the computation for the second stage. Now given a set of candidate matches (labels) for each subtemplate, the second stage will find a unique match for each subtemplate(including matching it to nothing) to form the best consistent labeling based on contextual information. We make use of DP here to do the job. Weighting factors are assigned to each subtemplate to reflect their relative importance. Subtemplates which are distinctive among others are given high weighting factors. Distance of the objects from the camera is assumed known. If the viewing angle relative to the normal of the object is given, the subtemplates may be transformed and scaled accordingly before matching with the boundary image. Details of the algorithm will be discussed in the following sections.

2.2 Subtemplate Matching

Before actual recognition can be done, a library of subtemplates for all objects to be recognized has to be formed in the first place. To do this, each object to be recognized is placed under ideal condition before a TV camera. A digitized image of which is acquired using a digitizer and is thresholded to produce a bi-level picture. External as well as internal boundaries are then extracted using conventional left-looking procedures[13]. For each boundary a set of subtemplates covering the entire boundary is created and stored in the library. Fig.2.1 shows such a tree-structured library. The leaves of the tree correspond to the basic elements for matching, whereas the structure itself contains the contextual information for constraint application. For a boundary template T of length L pixels, as many as L overlapping subtemplates may be used. On the other hand, one may use as few as approximately L/ℓ subtemplates, which are completely non-overlapping, where ℓ is the length of each subtemplate and varies from boundary to boundary depending on the size of the object. The degree of overlapping determines the number of subtemplates used and can be used in an hierarchical manner if desired. In our experiments the length of subtemplates varies from 10 to 30 pixels and is application dependent. Long subtemplate results in more confidence once it is successfully matched whereas shorter ones can handle severe occlusion and visually noisy images. Completely non-overlapping subtemplates are used throughout our experiments to save computing time.

A θ-s representation [8,14] for object boundary is used here to reduce computing effort. Along with each boundary template a θ-s representation is constructed and stored where θ is the angle of the slope of the boundary as a function of arc-length s. The advantage of using such a representation lies in the fact that rotation by θ_c of the boundary in the x-y space corresponds merely to a shift by θ_c in the θ-s space. Associating with each subtemplate a weighting factor which reflects the relative importance of the subtemplate is calculated and stored(details can be found in later sections). A reference point is chosen for each object. For each of the subtemplates of the object, a vector pointing to this point from the head of the subtemplate is defined as shown in Fig.2.2. Instead of using this vector to point to the accumulator to be incremented to implement Hough Transform, we use it in the evaluation of the matching result. The reference point is usually taken as the centroid unless there is an obvious convenient point. Fig.2.2 shows an object template with 10 subtemplates. Sub-template 3 is shown with its vector $\vec{v}$ pointing towards the reference point of the object. Fig.2.3a shows the same template with its θ-s representation.

During matching, boundary segments are extracted from the scene to be analysed. This can be done in a way similar to that used in constructing the library. For more general cases typical edge detection and chaining method should be used to produce useful segments for matching. Fig.2.3b shows an image boundary corresponding to a scene of two overlapping keys. All segments

with length greater than a certain value(length of the subtemplates) are used for matching. Each subtemplate is moved along the boundary segments one by one and matching is performed in the θ-s space by rotating the subtemplate to align with a section of a boundary segment. For convenience, we call the section as an object segment for later reference. An object segment is a candidate for matching with a subtemplate and is of the same length as the subtemplate. Let τ and β denote the subtemplate and boundary segment being matched respectively. Their corresponding θ-s representations are denoted by $\theta_\tau(s)$ and $\theta_\beta(s)$. The matching error between τ and the object segment located at pixel k along β is given by

$$\gamma_k = \sum_{s=1}^{\ell} [\theta_\beta(k+s) - (\theta_\tau(s) + \Delta\theta_k)]^2 \qquad (2.1)$$

$$\text{where } \Delta\theta_k = \frac{1}{\ell}\sum_{s=1}^{\ell}\theta_\beta(k+s) - \frac{1}{\ell}\sum_{s=1}^{\ell}\theta_\tau(s) \qquad (2.2)$$

is the difference between the average slopes of the object segment and the subtemplate.

This is repeated for all boundary segments. A matching position where the matching error is smaller than a threshold is entered into a match table. The matching error is another entry. The corresponding angle of rotation for the subtemplate to align with the object segment at that position is also recorded in the table. If more than m such matches occur for a subtemplate, only the best m matches which give the smallest matching errors are retained. Hence the size of the table is limited to n x m, where

n is the number of subtemplates used for matching. The above procedure is repeated for all subtemplates. Fig.2.3c illustrates the matching process. Fig.2.3d shows the result of the matching. Note that the best match is not necessarily the correct one. When all have been completed, we go on to the second stage.

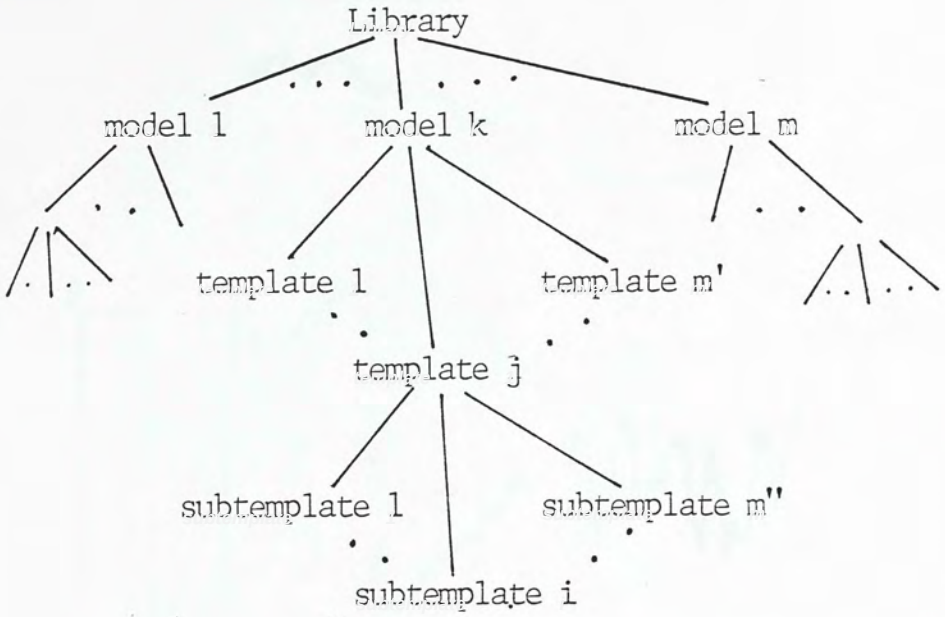


Fig 2.1 A tree-structured library

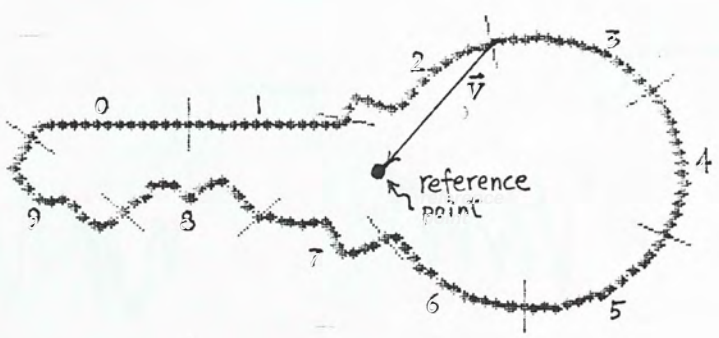


Fig 2.2 A template with 10 subtemplates

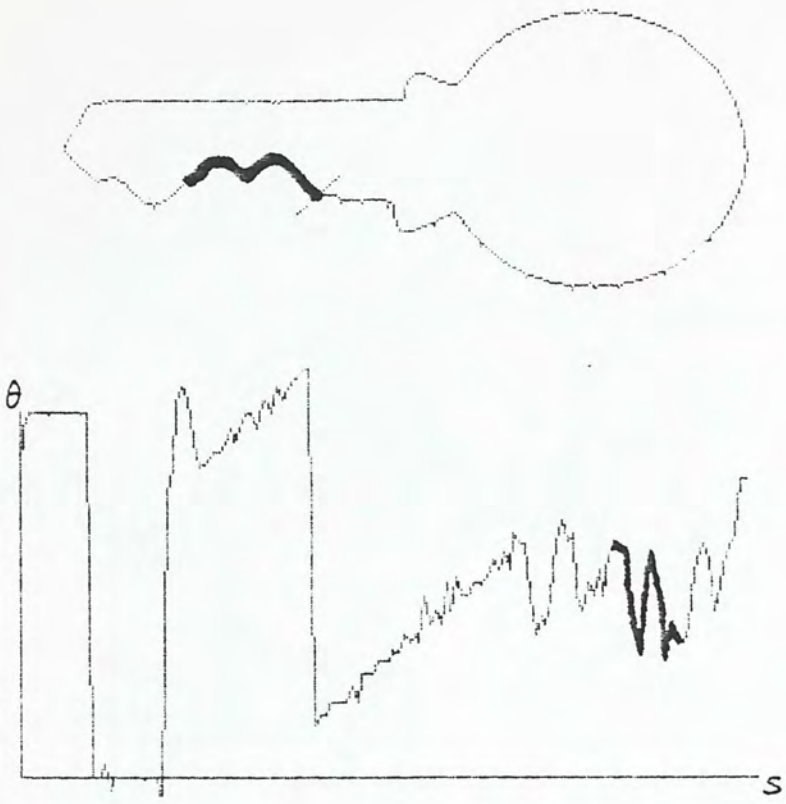


Fig 2.3a A Template & its θ-s Representation
The heavy line show a typical subtemplate
& its θ-s correspondent

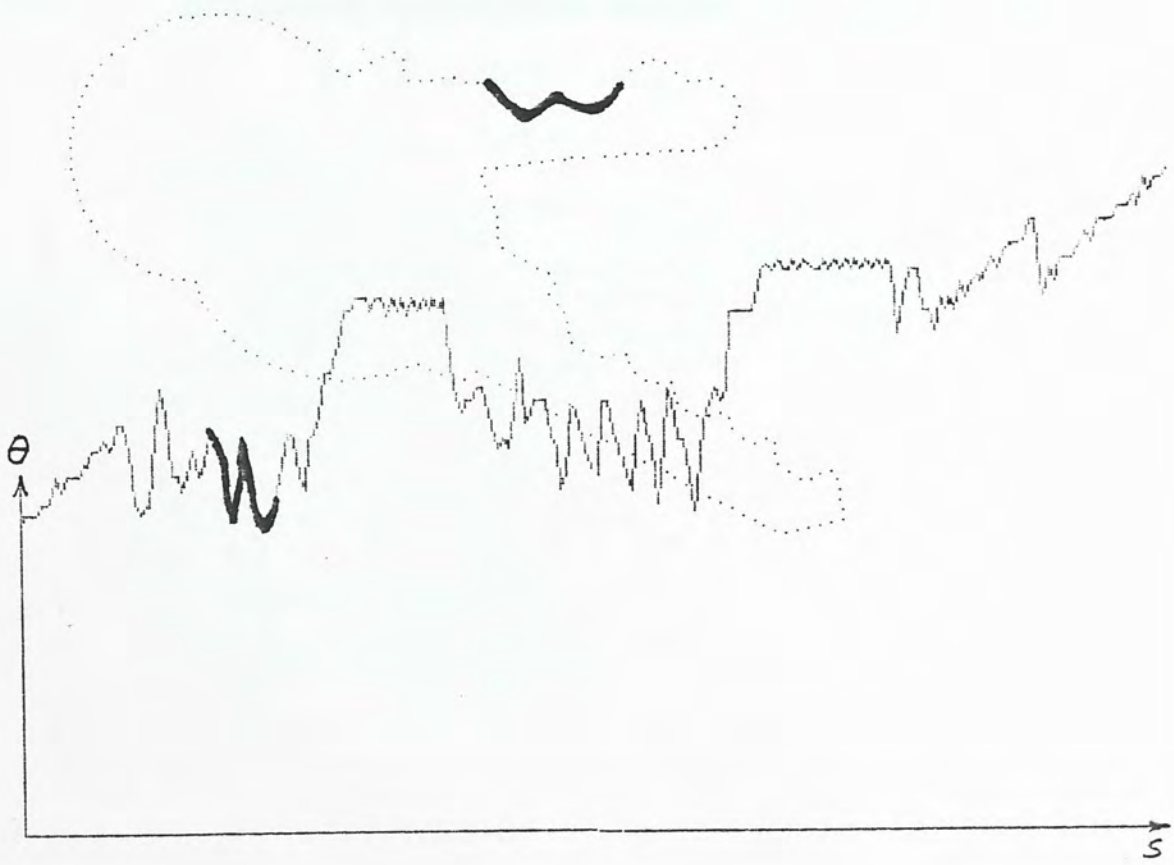


Fig 2.3b An image boundary & its θ-s representation
The heavy lines correspond to the subtemplate
in Fig.3a.

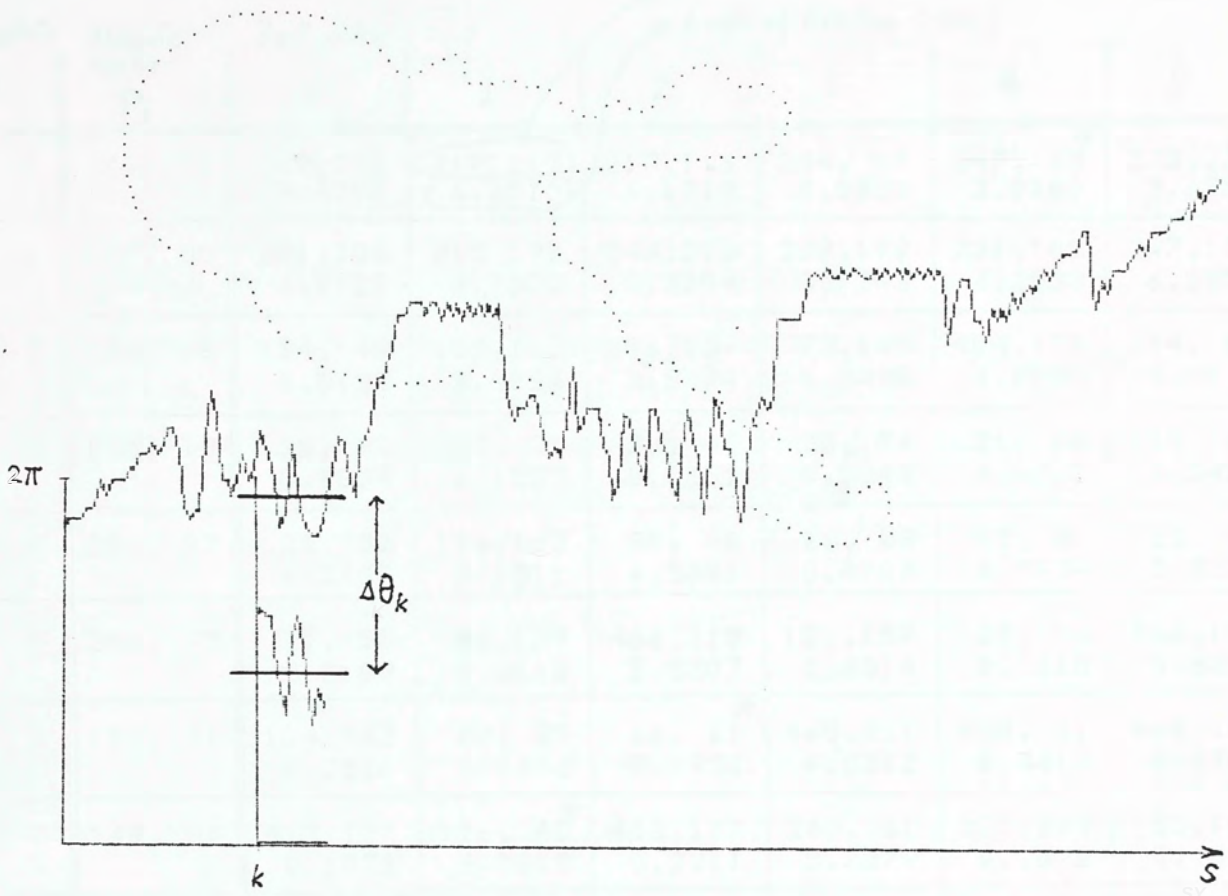


Fig. 2.3c The subtemplate matching process

Subtemplate i	Absolute Position P_i	Best match ∅	1	2	3	4	5
0	24, 55	269,222 3.6790	219,112 6.2510	217,116 6.1719	244, 69 3.0533	247, 65 * 3.0100	273,221 3.6284
1	79, 50	271,105 6.2727	215,195 3.7372	246,213 3.7204	223,199 3.7243	231,108 6.2552	247,107 6.2552
2	134, 48	124, 40 0.8122	159,160 3.5884	306,139 3.5694	72,143 4.3499	404,178 4.0958	244, 69 3.4575
3	185, 18	38, 33 5.9034	50, 25 6.1535	30, 40 5.7351	23, 96 4.5348	21, 88 4.6922	108,162 3.5425
4	241, 37	23, 52 4.2737	116,162 2.1511	30, 40 4.5331	21, 88 3.4902	42, 30 4.7934	23, 96 3.3329
5	246, 93	21, 88 2.3188	46,120 1.6013	466,115 5.5507	131,159 0.6514	23, 96 2.1615	466,123 5.6859
6	197,121	104,162 0.2510	62, 22 3.0606	66, 21 * 3.0450	465,111 4.5312	435, 61 3.7612	464,103 4.4245
7	148, 94	412,177 0.1879	116, 45 * 3.0315	408,177 0.2211	248, 61 5.7279	277,220 0.1072	83,153 0.4623
8	104, 86	161, 48 * 2.9702	324,145 0.0101	157, 48 3.0161	179,172 0.1291	183,175 0.1750	175,170 0.0785
9	56, 84	208, 47 * 2.9732	211, 43 3.2078	240,137 0.5155	284,210 5.4523	205, 51 2.7488	464,135 4.7677

Matching Position (x,y)

Angle of Rotation (rad)

Fig 2.3d The match table.
Each entry (i,j) corresponds to the jth best match of subtemplate i. The first two items in each entry correspond to the matching position whereas the remaining item corresponds to the angle of rotation for alignment. Entries marked with a '*' correspond to the correct matches. Clearly the best match is not necessarily the correct one.

2.3 Selecting the Best Set of Matches using Dynamic Programming

As a result of the first stage of the algorithm, we have a match table consisting matches of the form M_{ij} : the jth matching for subtemplate i. Each entry (i,j) contains the following items:

X_{ij} : matching position - where that match occurs,
γ_{ij} : matching error - how well does it match,
θ_{ij} : angle of rotation - angle by which the subtemplate must be rotated to align with the jth object segment.

The problem here is in some way similar to the conventional segmentation-labeling problem. Given a set of subtemplates (model segments) and a set of object segments, each subtemplate should be labeled either as one of the object segments or as not belonging to the object. Unlike the conventional problem, each subtemplate is now restricted to a small set of possible labels, and hence the complexity involved is somewhat reduced. It is possible to use dynamic programing(DP) to solve this problem. Alternatively the subtemplate matching process may be viewed as a model-guided segmentation. The process will assign object segments with initial probabilities of being associated with a certain object subtemplate. One may design a relaxation scheme in the conventional manner to solve this problem, but it will be much less efficient than DP.

Dynamic Programming(Bellman and Dreyfus 1962,[15]) is a technique for solving optimization problems when not all variables in the evaluation function are interrelated simultaneously. Consider the problem

$$\max_{x_1} h(x_1, x_2, x_3, x_4) \tag{2.3}$$

If nothing is known about h, the only technique that guarantees a global maximum is exhaustive enumeration of all combinations of discrete values of $x_1,\ldots,x_4$. Supose that

$$h(.) = h_1(x_1,x_2) + h_2(x_2,x_3) + h_3(x_3,x_4) \tag{2.4}$$

For each x_2 in $h_1(x_1,x_2)$ maximize h_1 over x_1:

$$f_1(x_2) = \max_{x_1} h_1(x_1,x_2) \tag{2.5}$$

Since the value of h_2 and h_3 do not depend on x_1, they need not be considered here. Continue this way and eliminate x_2 by computing $f_2(x_3)$ as

$$f_2(x_3) = \max_{x_2}[f_1(x_2) + h_2(x_2,x_3)] \tag{2.6}$$

$$f_3(x_4) = \max_{x_3}[f_2(x_3) + h_3(x_3,x_4)] \tag{2.7}$$

and finally

$$\max_{x_1} h = \max_{x_4} f_3(x_4) \tag{2.8}$$

By then the value of x_4 that maximize h is known. We then trace back to obtain the value of x_3 that maximaize f_3, and so on until finally we come out with a set of labeling $\{x_i\}$ which maximize h.

Generalizing to n variables, we have

$$f_{k-1}(x_k) = \max_{x_{k-1}}[f_{k-2}(x_{k-1}) + h_{k-1}(x_{k-1},x_k)] \tag{2.9}$$

$$\max_{x_1} h(x_1,\ldots x_n) = \max_{x_n} f_{n-1(x_n)} \tag{2.10}$$

Ballard[11] showed that if each x_i takes m discrete values, then to compute $f_n(x_{n+1})$ one must evaluate the maximand for m different combinations of x_n and x_{n+1}, so that the computational effort involves $(n-1)m^2+m$ such evaluations, which is a great improvement over exhaustive evaluation that involve m^n evaluation of h.

DP has been quite useful in certain aspects. Ballard and Sklansky[12] used the algorithm in finding a "best" representation for the boundary of a region of circularity in the context of recognizing tumors in chest radiographs.

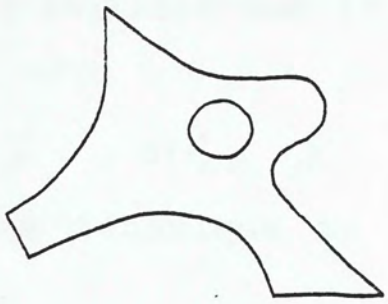
Despite its effectiveness, the assumption that not all the variables in h are interrelated simultaneously is not quite true in most cases. However the technique is still very useful if we are ready to give up a few extrodinary occasions such as the one depicted in fig.2.4. Since each unoccluded subtemplates are on their own, the DP scheme fails. Anyhow the situation can be improved if smaller subtemplates are used. After all, such a situation seldom occurs.

Now let's get back to our problem. Given a set of subtemplate $\{\tau_1, \tau_2 \ldots \tau_n\}$, each τ_i can have m different labels $\{M_{11}, M_{12} \ldots M_{1m}\}$. We try to select for each τ_i a unique label such that the compatability c of the resulting label set

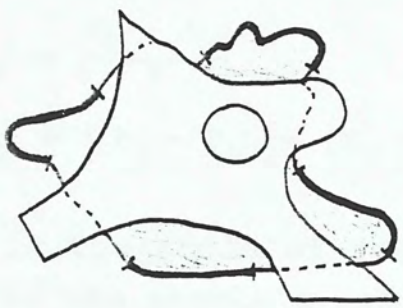
$$c = c(\tau_1, \tau_2 \ldots \tau_n) \qquad (2.11)$$

香港中文大学图书馆藏书

a. An object & its subtemplates



b. Another object



c. The apparent scene

Fig 2.4 A special case which the DP scheme might fail to handle. The heavy lines corresponds to the matched subtemplates, since they are not connected, the DP scheme fails if there exist potential matches(false matches for all of the occluded subtemplates.

is maximized. One immediately realized that if we hypothesize c to be of the form

$$c = c(\tau_1, \tau_2) + c(\tau_2, \tau_3) + \ldots c(\tau_{n-1}, \tau_n) \qquad (2.12)$$

then we can used the above technique to solve the same optimization problem.

To incorporate the DP scheme in the second stage of the algorithm, first define two metrics d_1 and d_2 between two matches $(M_{ij}, M_{k\ell})$ which measures their difference:

$$d_1(M_{ij}, M_{k\ell}) = \left|\frac{\theta_{ij} - \theta_{k\ell}}{\pi}\right| \qquad (2.13)$$

$$d_2(M_{ij}, M_{k\ell}) = \left|(P_i - P_k) - (X_{ij} - X_{k\ell})\right| \qquad (2.14)$$

where θ_{ij} is the angle of rotation for alignment in M_{ij},
p_i is the absolute position of τ_i,
X_{ij} is the matching position of M_{ij}.

Note that d_1 corresponds to a rotational metric whereas d_2 corresponds to a positional one. The compatibility c between two matches is thus defined as:

$$c(M_{ij}, M_{k\ell}) = r\,(1 - d_1) + \frac{(1 - r)}{(1 + r'd_2)} \qquad (2.15)$$

where r is a single constant used to adjust the relative weight of the two factors. r' is a constant factor depending on the resolution of the image.

The reason behind this is simple. If two matches are compatible, their subtemplate should have approximately the same angle of rotation and their relative displacement should be about the same as that of the corresponding object segments.

Now given the set of all M_{ij}'s, i=1 to n where n is the number of subtemplates in the match table, we calculate

$$
\begin{aligned}
&1. \quad f_1(x_2) = \max_{x_1} [c(M_{1x_1}, M_{2x_2})] \\
&2. \quad f_2(x_3) = \max_{x_2} [f_1(x_2) + c(M_{2x_2}, M_{3x_3})] \\
&\quad \vdots \qquad\qquad \vdots \\
&k. \quad f_k(x_{k+1}) = \max_{x_k} [f_{k-1}(x_k) + c(M_{kx_k}, M_{k+1,x_{k+1}})] \qquad (2.16) \\
&\quad \vdots \qquad\qquad \vdots \\
&n\text{-}1. \quad f_{n-1}(x_n) = \max_{x_{n-1}} [f_{n-2}(x_{n-1}) + c(M_{n-1,x_{n-1}}, M_{nx_n})] \\
&n. \quad f_n = \max_{x_n} f_{n-1}(x_n)
\end{aligned}
$$

where $f_{k-1}(x_k)$ represents the optimal compatibility for the matchings corresponding to subtemplates 1, 2, . . . k if subtemplate k is assumed to be matched with x_k, a segment of the boundary image of the same length.

Let τ_i and x_j denote subtemplate i and object segment j respectively. Step 1 gives the most compatible match for τ_1 given that τ_2 is matched to a given x_2. This is repeated for all x_2. Hence if x_2 is determined, x_1 is determined also. Step 2 gives the most compatible match for τ_2 given that τ_3 is matched to a given x_3 and is repeated for all x_3. Again x_2 can be determined once x_3 is fixed. We go on this way until finally at step n we find the most compatible match for τ_n. Then, we trace back to obtain the best set of x_i's and M_{ij}'s.

When the image segment corresponding to a subtemplate is occluded or missing in the scene, no good match should be found. The subtemplate is assumed to be matched to a nil-segment. Therefore we associate with each subtemplate a nil match, and the compatibility for such matches are given by

$$c(M_{ij}, M_{k\ell}) = \begin{cases} c_1(1 - e^{-\frac{x}{a_1}}), & j=\text{nil or } \ell=\text{nil} \\ c_2 + (1 - c_2)e^{-\frac{x}{a_2}}, & j=\ell=\text{nil} \end{cases} \quad (2.17)$$

where $X = |P_i - P_k|$ is the absolute distance between the two subtemplates; c_1, c_2, a_1, a_2 are adjustable parameters.

Fig.2.5 plots c vs X for both cases involving nil matches. The idea is that if one subtemplate is being occluded, it is likely that a nearby subtemplate is also being occluded. The four parameters c_1, c_2, a_1 and a_2 are the key parameters which affect the outcome of the algorithm. A high c_1 and c_2 with low a_1 and high a_2 favour the nil matches. If c_1 and c_2 are set too high, it may cause a trivial solution with all subtemplates labeled nil. Too low a a_1 and a_2 with high c_1 and low c_2 may cause incompatible matches to appear in the final assignment. Our strategy here is to give a good initial value for c_1 and c_2(both 0.8 in our experiments) and fix a_1 and a_2 (both equal 20 in our experiments). After the DP is applied, an evalutaion is done on the resulting assignment. The average and standard deviation of the angle of rotation for all subtemplates within the assignment are computed. Each subtemplate selected by the DP above is rotated and the point to which the subtemplate vector (defined by the subtemplate and the

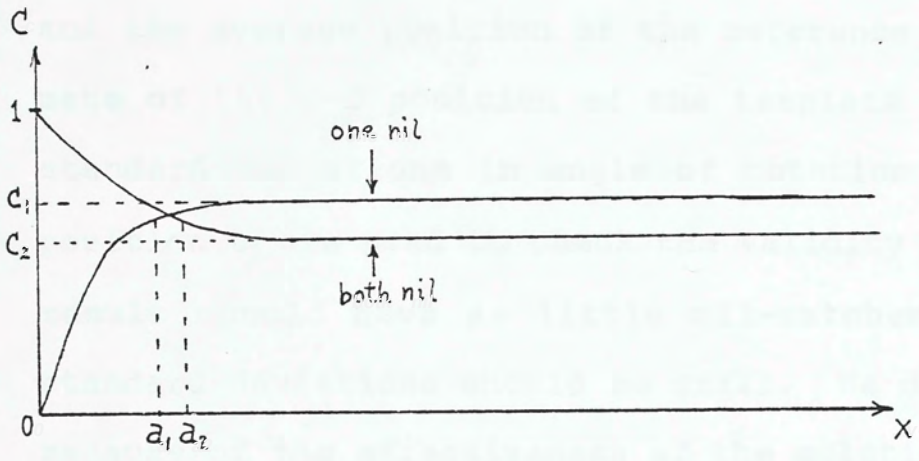


Fig 2.5 Compatability function for nil-class
In practice $a_1=a_2=20$, c_1 & c_2 are initially set to 0.8 & are dynamically adjusted during the second stage.

reference point) points is recorded. If the matchings are all accurate and correct the location of this point should be the same for all subtemplates. Hence we estimate the location by averaging over all subtemplates. The average angle of rotation and the average position of the reference point provide an estimate of the 2-D position of the template to be recognized. The standard deviations in angle of rotation σ_1 and reference point position σ_2 are used to check the validity of the solution. A good result should have as little nil-matches as possible, and the standard deviations should be small. We define a quantity η as a measure of the effectiveness of the solution as follows:

$$\eta = (1 - e^{-\frac{n'}{a_3 n}})(e^{-(b_1\sigma_1 + b_2\sigma_2)}) \qquad (2.18)$$

where n is the number of subtemplates used in the matching, n' is the number of subtemplates successfully labeled in the final stage; b_1, b_2 and a_3 (typical values are 11, 0.1, 0.3 respectively) are adjustable parameters.

If the computed confidence η is above a predefined threshold T_η we accept the solution and stop. On the other hand if η is low, attempt is then made to get better results by modifying the parameters. If it turns out that all matches are nil, c_1 and c_2 will be lowered and the process is repeated. If on the other hand the standard deviations are greater than the predefined values, c_1 and c_2 are raised and the process is again repeated. In our experiments the process is allowed to be repeated for at most 4 times. If by then the result is still no good, we claim that

the object to be recognized is not present in the scene. In practice if an object does exist in the scene, the process usually gives the correct result within 1 or 2 trials. Fig.2.6 illustrates the above in the form of a block diagram. Fig.2.7 shows the result of applying the algorithm to the problem of overlapping keys in Fig.2.3. The key to be detected is correctly identified.

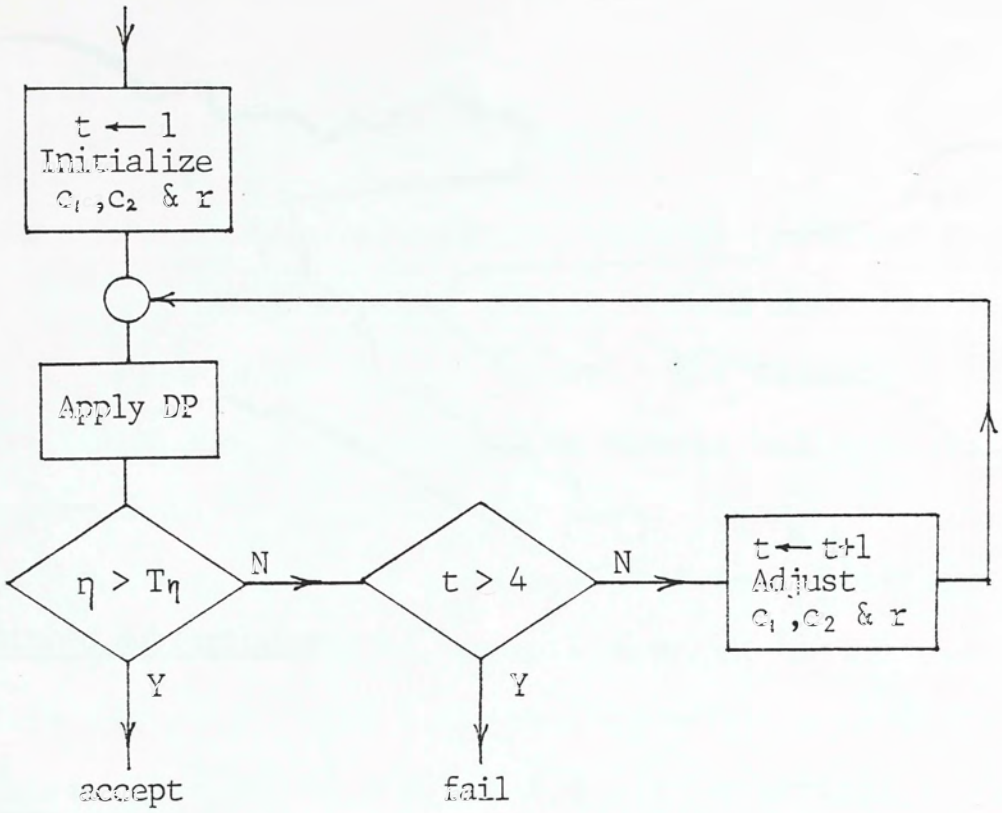
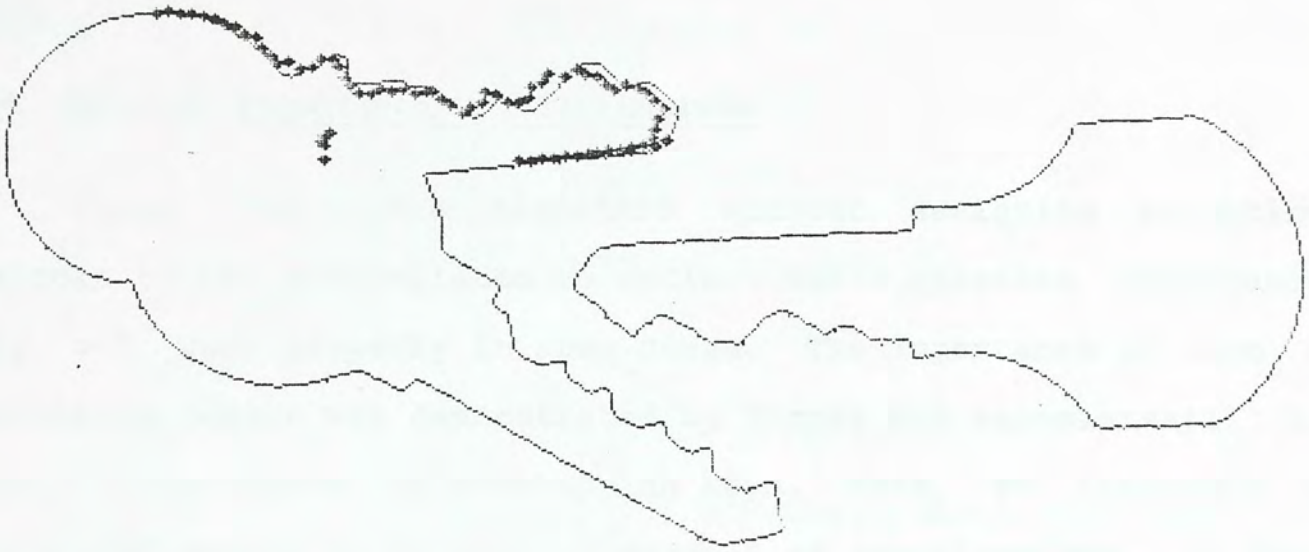
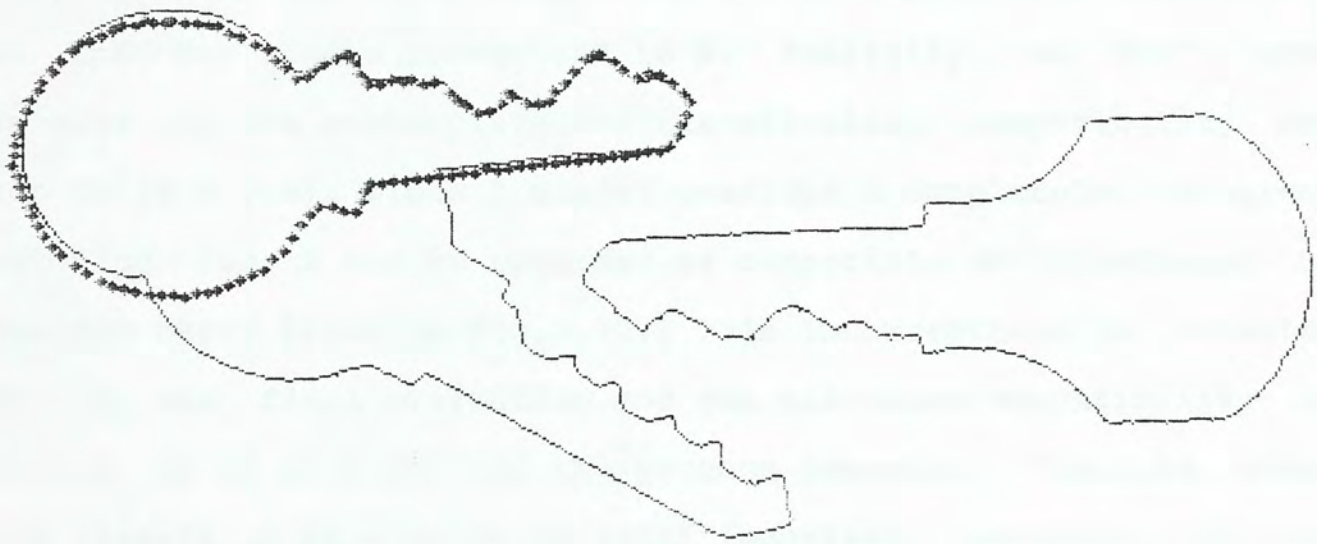


Fig 2.6 Application of Dynamic Programming



a. Matched subtemplates



b. Final result

Fig 2.7 Result of identification with $r=0.5$, $c_1=c_2=0.8$.
Total no. of subtemplates $n=10$
No. of matched subtemplates $n'=5$
Mean rotation = 3.0060 $\sigma_1=0.0302$
Mean ref.point = (127,70) $\sigma_2=4.123$
$\eta = 0.380$

2.4 Relative Importance of Subtemplates

Using the above algorithm without assigning weighting factors to the subtemplates to reflect their relative importance may not work properly in some cases. The importance of such a weighting scheme was demonstrated by Turney and associates[11] in their experiments on overlapping keys. Here, we introduce a weighting scheme to be used at stage 2 of our algorithm. We then test the algorithm using an example similar to that of [11] as depicted in Fig.2.9. Fig.2.9b shows a scene consisting of three very similar looking keys A, B & C with B partially occluded by A. The key to be recognized is B. Initially, we don't give weights to the subtemplate and the nil-class compatibility are set to (0.8,0.8). Since A almost overlaps B completely, incorrect matching for B can be regarded as compatible as illustrated by the two heavy lines in Fig.2.9c. This incorrectness is detected during the final evaluation and the nil-class compatibility is raised to (0.85,0.85) and the process repeated. Fig.2.9d shows the result of this which is still incorrect. According to the algorithm, the heavy line on the right side is more compatible than that on the left side(this is reflected in the scattering of the reference points of the left subtemplates in contrast to the concentration to one single point for the right-handed ones). To solve this problem we assign weighting factors to each subtemplate as described by the following section. For a subtemplate which is very distinctive among the other subtemplates from the same model or from other models, we give it a high weighting

factor. We adjust the compatibility of each pair of matches by the weighting factors of the two subtemplates involved. That is in the kth step of the DP, we calculate

$$f_k(x_{k+1}) = \max_{x_k} [f_{k-1}(x_k) + \mu w'_k w'_{k+1} c(M_{kx_k}, M_{k+1,x_{k+1}}) + \xi(\frac{W_k}{1 + \gamma_{k,x_k}} + \frac{W_{k+1}}{1 + \gamma_{k+1,x_{k+1}}})] \quad (2.19)$$

where W_k' is derived from the weighting factor of subtemplate k and is a quantity within the range $(1-\Delta, 1+\Delta)$, and Δ is in the range $(0,1)$. μ and ξ are adjustable parameters.

In our experiments, the algorithm seems to work well with μ large and ξ small. That means the effect of matching error does not count in the second stage. The actual influence that the fitness of matching of a subtemplate on telling whether it is the correct one should depends on one hand on the importance of the subtemplate and on the other hand the threshold for accepting entries in the match table. As a matter of fact when an entry is entered into the match table, we already assume that it is good enough to be a potiential correct match. Again one should be reminded that the best match is not necessarily the correct match. Since both the two issues are already taken into consideration, one in the weighting scheme and the other in the thresholding strategy in building the match table, setting ξ to zero is reasonable. The choice of Δ depends on how much influence the weighting scheme should have in a particular application. To be more accurate, we adjust the value of Δ adaptively according to the performance of the algorithm.

Starting with an initial value of Δ equals 0.1, we compute a solution using the set of W_k's for this value of Δ. If the average of the weighting factors of the finally matched subtemplates(non-nil), δ, is below a certain threshold T_δ(1/n is used in our experiments), this indicates that the result is not significant enough in distinguishing the object to be detected even if $\eta > T_\eta$. We should then repeat the DP with Δ incremented by 0.1 to make the weighting scheme more influential. Now, our algorithm in the second stage is modified as follows.

1. Initialize all adjustable parameters.
2. Use the new DP scheme with weighting to select a consistent solution from the data in the match table.
3. If η is above the prescribed threshold T_η, go to 4. Otherwise, go to 2, with c_1 and c_2 modified as described in section 2.2 unless this has already been done four times. In the latter case, go to 6.
4. If δ is above the prescribed threshold T_δ, the object is detected. Stop and exit. Otherwise go to 5.
5. If Δ equals 0.5 already, Go to 6. Otherwise go to 1 with Δ incremented by 0.1 in 1.
6. Object is not in the scene. Stop and Exit.

The block diagram of the above algorithm is shown in Fig.2.8. Fig.2.9e and fig.2.9f show the result after the algorithm is modified to make use of the weighting scheme as described above. The effect of weighting the subtemplates gives us a correct solution.

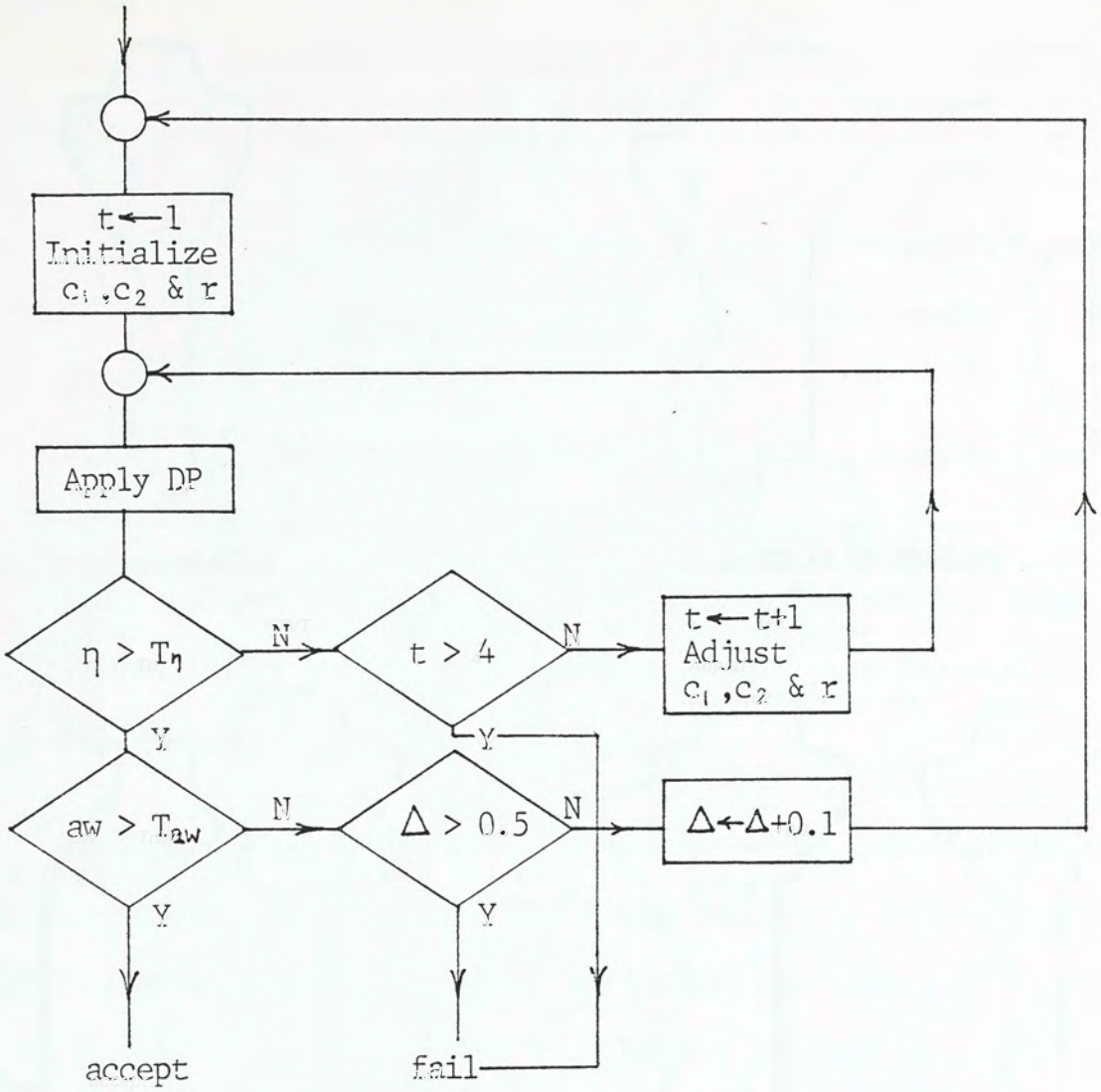
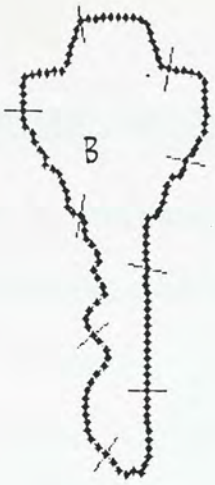
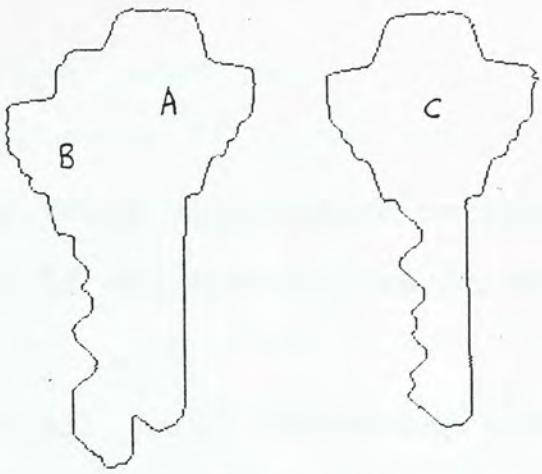


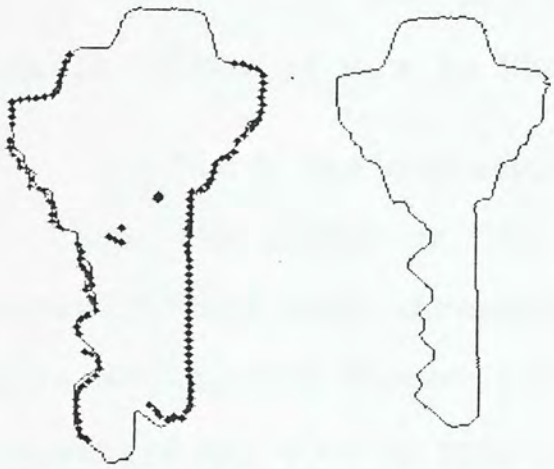
Fig 2.8 Modification of the 2nd stage to make use of subtemplate weighting



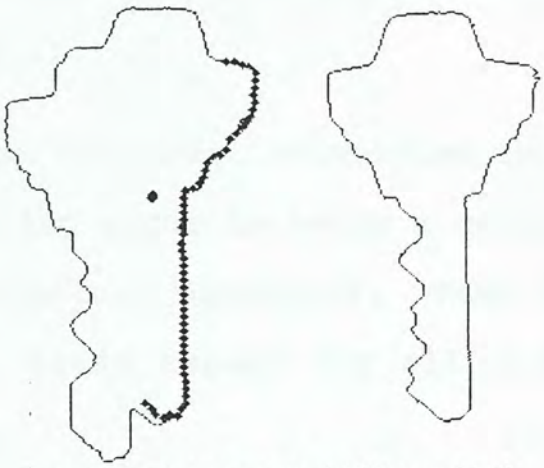
a. Key to be recognized



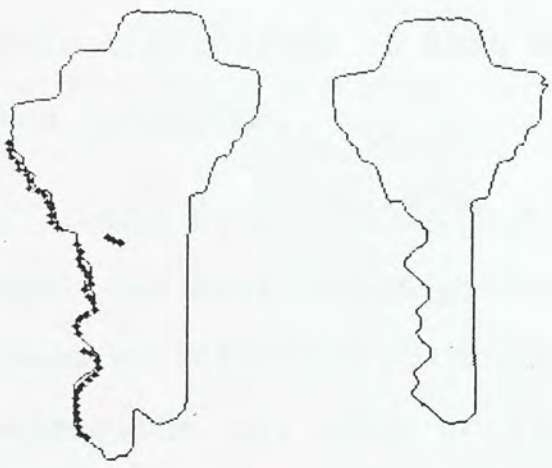
b. Scene to be analysed



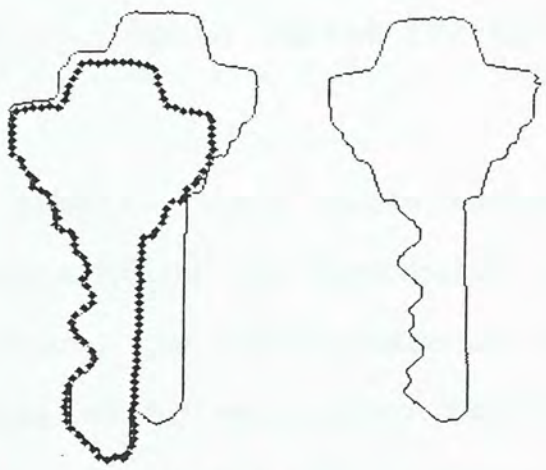
c. Incorrect result due to low nil-class compatability
$c_1 = c_2 = 0.8$, $\Delta = 0$
$\eta = 0.123$, $aw = 0.119$



d. Nil-class compatability raised
$c_1 = c_2 = 0.85$, $\Delta = 0$
$\eta = 0.430$, $aw = 0.043$



e. Application of weighting adjustment
$c_1 = c_2 = 0.8$, $\Delta = 0.20$
$\eta = 0.337$, $aw = 0.205$



f. Result

Fig 2.9 Application of the weighting scheme

2.5 Calculation of Subtemplate Weights

Given a subtemplate τ_i, we propose three alternative methods for calculating its weighting factor w_i if all the objects in the image are known.

1. Match the subtemplate against all model boundaries just as if they were ordinary object segments extracted from the scene to be analysed. Take a summation over all the matching errors to give w_i. Repeat for all other subtemplates and normalize the resulting set of w_i's to the range (0,1).

2. Match the subtemplate against all model boundaries as in 1. Count the number of time the matching error is below a certain threshold(the same threshold used in actual matching). Take the inverse of that number to give w_i. Again repeat for all i and normalize all w_i's to the range (0,1).

3. A combination of 1 and 2. Take a summation over the quantity $1/(1+\gamma_k)$ for those γ_k which fall below the threshold. Take the inverse of that sum to give w_i. Again repeat for all i and normalize.

The motivation for 2 and 3 is simple. Only those matches that are good enough(with matching error below the threshold) are used to infer the solution. In practice, the performance of the algorithm are more or less the same using weighting factors computed by any of the three methods. However 2 and 3 seems to give a little bit better performance than 1 in our experiments.

If the scene to be analysed is known beforehand to consist of many unknown objects that do not belong to the model set, the weighting factor should be composed of two components. The first component is computed by any one of the above methods using all known objects expected to be in the image. In case if only the object to be detected itself is known, this component is computed using only this object. The design of the second component is based on a conjecture we made on the statistical nature of the boundary curve. We hypothesize that the more rugged is the object segment, the more unlikely is it similar to other object segments. This second component w_i'' can be computed as

$$\hat{w}_i = \sum_{k=1}^{\ell} |\Delta\theta_k| \tag{2.20}$$

$$w = \max_i \hat{w}_i \tag{2.21}$$

$$w_i'' = \frac{\hat{w}_i}{w} \tag{2.22}$$

$$\tilde{w}_i = \alpha_i w_i + (1 - \alpha_i) w_i'' \tag{2.23}$$

$$\alpha_i = \frac{N_k}{N_t} \tag{2.24}$$

where $\Delta\theta_k$ is the change of angle of slope at pixel k, N_k is the number of known objects expected in the scene and N_t is the total number of objects in the scene; $\tilde{w}_i$'s are used instead of w_i's as the weighting factors for the subtemplates.

2.6 Experimental Results and Discussions

Several tests are performed to verify the effectiveness of the algorithm. Images of object scenes in our experiments are acquired using a TV camera with a Colorado digitizer. Fig.2.9 shows an experiment with three very similar looking keys. Another experiment with some commonly used tools is performed and is illustrated in Fig.2.10. The object to be detected is correctly located. Fig.2.11 shows an experiment with 12 parts. Our algorithm seems to work well in all these cases.

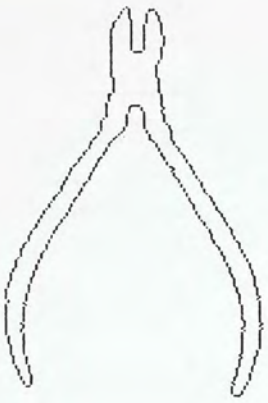
The algorithm is implemented on an IBM PC/XT with 8087 co-processor and is coded in Pascal. Depending on the complexity of the scene, the typical time for locating an individual object in our experiments is about 1-2 minutes. The experiment with the keys in Fig.2.3 takes less than one minute. To locate the parts in Fig.2.11 takes 1.5-2 mins on an XT, with 90% of the time spent on subtemplate matching. It takes less than one minute to run on an IBM PC/AT.

Creation of the model library is an interactive process. Image of the model is acquired and processed under human supervision. The time required to compute the weights for the library in Fig.2.10 takes an hour on an IBM PC/XT, and about half an hour on an IBM PC/AT. The feature extraction process can be made automatic without human intervention by letting the machine to choose a subtemplate set by varying subtemplate size and location such that the overall weighting of the set is maximum, ie. each

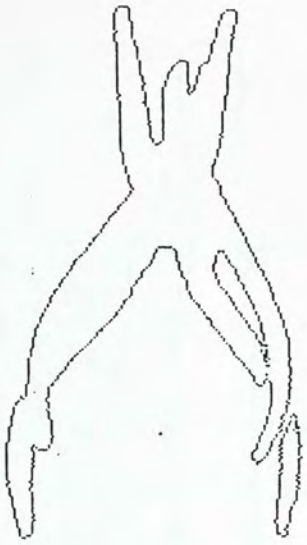
subtemplate chosen should be as distinguishable from others as possible. In fact this is one possible direction of extending the work to a fully automized scheme for two dimensional shape recognition.

One obvious trick that greatly reduces the overall recognition time is to reduce the number of subtemplates that is being matched, since the subtemplate matching process is the most time-consuming part of the whole recognition process. We may limit our attention to those subtemplates with high enough weightings for matching. In our experience, using only 60% of the subtemplates of an object still gives satisfactory results. This would mean a 40% reduction in computing time. As a matter of fact omitting those subtemplates that are very similar to others has the advantage that some problems associated with stage 2 of the algorithm can be eliminated, for example the situation depicted in fig 2.9 did not happen if we match only the important ones.

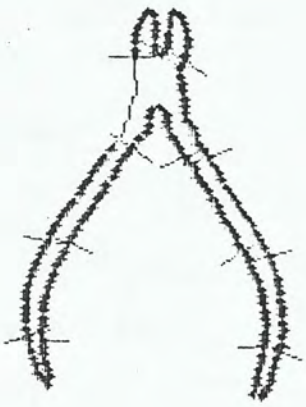
By using DP to select an optimal consistent set of matches, we have been able to use non-overlapping subtemplates to obtain a reliable solution. We don't need a large number of overlapping subtemplates to get the result in a statistical manner as in other schemes[11]. As most of the time (90%) is spent on subtemplate matchings this would make our algorithm ten times faster in many cases. Like other schemes, our algorithm may have trouble when important subtemplates are occluded. However this problem can often be solved with reduced subtemplate length.



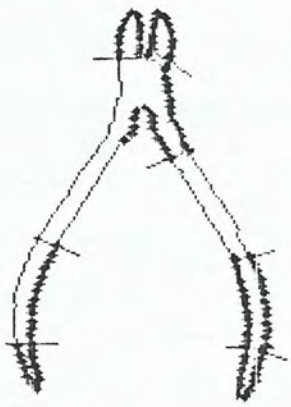
Model Template



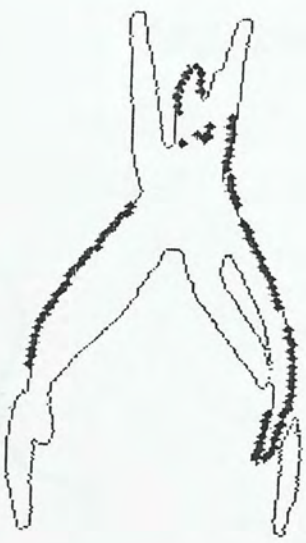
Apparent Object



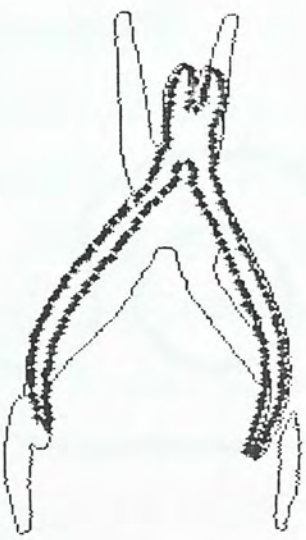
Subtemplates



Subtemplates with high weighting

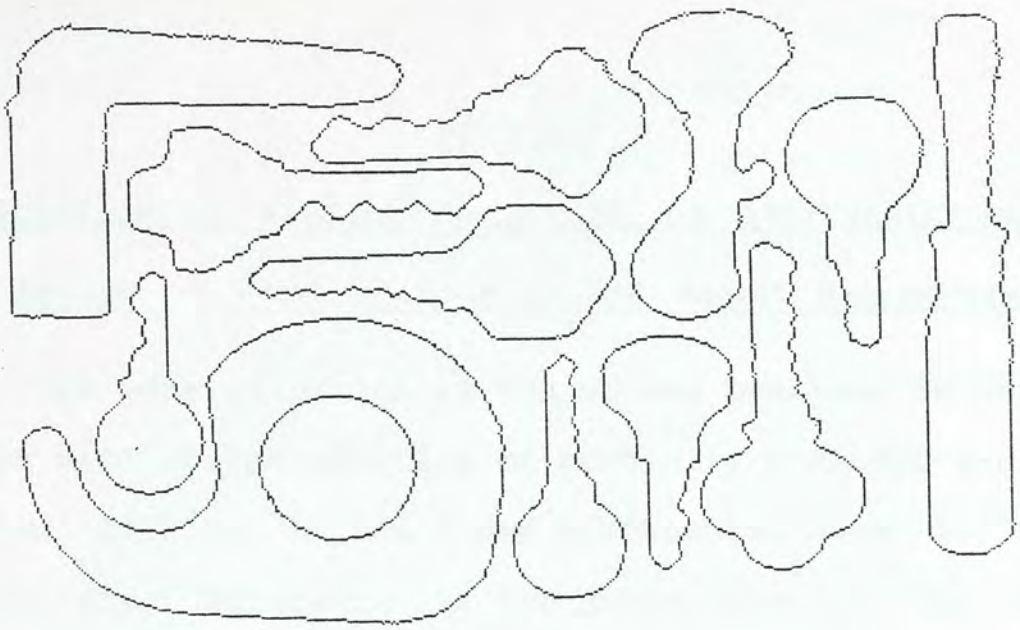


Matched subtemplates

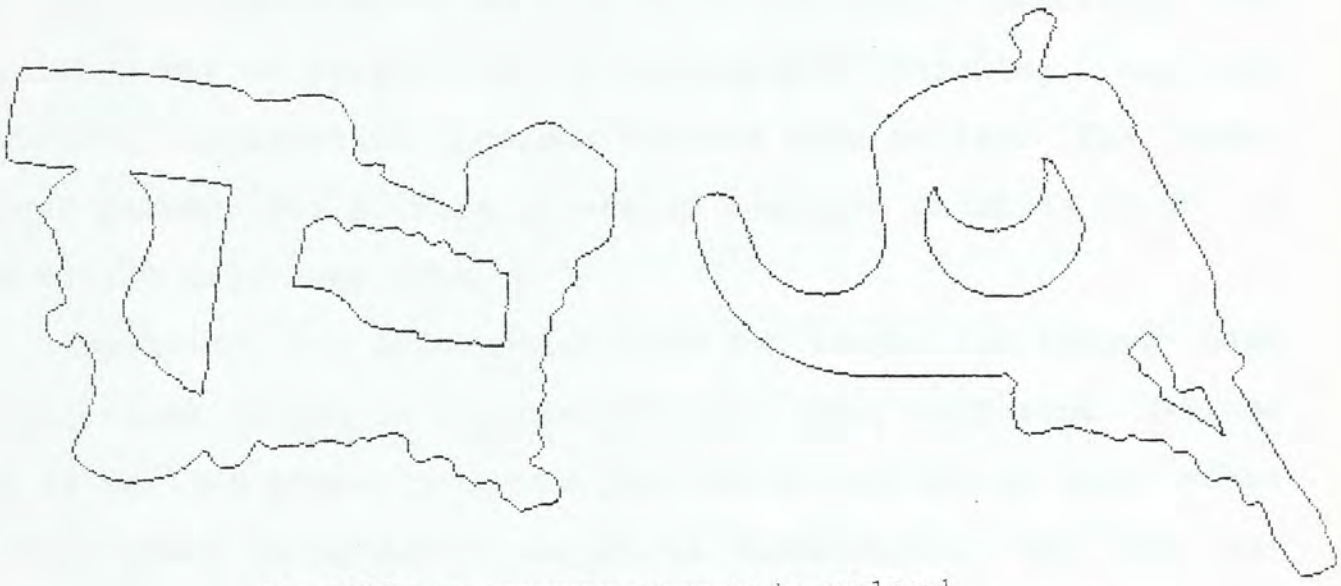


Result

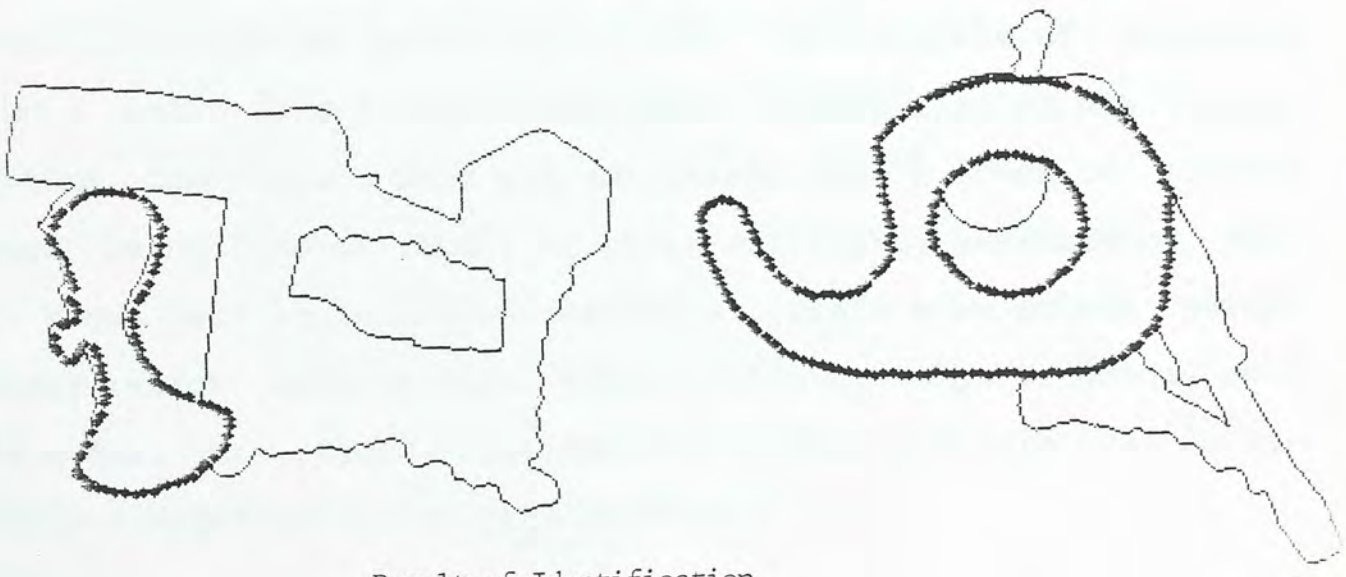
Fig 2.10 Another Experiment



A set of 12 parts



Two differnet apparent scene to be analysed



Result of Identification

Fig 2.11 An experiment with 12 parts

CHAPTER 3

RECOGNITION OF PARTIALLY-OCCLUDED 3-D ARBITRARILY-SHAPED OBJECTS

3.1 Review of Previous Work in 3-D Object Recognition

The generalization of the scheme proposed in Chapter 2 which deals with the recognition of partially occluded 2-D arbitrarily-shaped objects to the three dimensional case at first glance seems straightforward if 3-D information of the object under recognition is available. As a matter of fact such a generalization is straightforward as long as we can find a practical and efficient way of performing 3-D subtemplate matching, and the constraint application process remains more or less the same. However subtemplate or more generally template matching in 3D is more easily said than done.

Recovering 3-D information from 2-D images has always been an important issue in computer vision. Many different methods such as various stereo programs [16] which use two or more views of the scene at different angles to infer depth, and the so-called shape from XX techniques, such as shape from shading[17], shape from regular patterns[18] etc., are capable of producing fairly accurate and dense range maps. There also exists active ranging techniques that project energy onto a scene to measure range using time-of-flight or phase-difference measurement such as those used in laser range-finders. There also exists active triangulation methods which projects light strips or grids onto the scene. The readers are referred to Besl and Jain [19] pp.92-96 for a brief review on this subject.

Regarding the types of information utilized, existing algorithms in 3-D object representation and recognition can be broadly classified into three categories. One works on and only on 2-D data, treating the 3-D recognition problem using 2-D techniques. Others utilize 3-D information, which is either obtained via range-finders or recovered from 2-D intensity images. The third category uses both and forms a hybrid scheme.

Thus far the acquisition of 2-D intensity image is much easier, cheaper and faster than the acquisition of 3-D range data. For the recognition of 3-D objects with a small number of stable orientations, it is convenient and sufficient to creat a library of 2-D silhouettes or contour projections for representation and matching, as long as the silhouettes are distinctive enough. One example that used similar ideas is aircraft identification[20]. The silhouette of the craft seen at different viewing angles are recorded in the form of Fourier Descriptors. However as a general technique such schemes suffer from the overwhelming size of the library as well as the required time to search for a solution. Other techniques within the first category includes direct identification of certain simple geometric entities within a single intensity image. Cernuschi-Frias and Cooper[21] used reflectance maps to detect planes, cylinders and spheres in a single intensity view but the technique is rather restricted to be useful in the general case. There also exists rule-based system such as the ACRONYM system by Brooks et al.[22] which generates model-based 3-D interpretations of 2-D images. There are many other techniques in this category. For a more detailed review see Besl and Jain[19] pp. 7-

With the advent of range-finding techniques, there has been a surge in the development of methods for dealing with range data. The use of depth information in recognizing 3-D objects seems natural and justified since a single depth map conveys much more information than a single intensity map. Besl and Jain[19] showed that the intensity-image object-recognition problem is generally more difficult owing to an illumination-reflectance operator I which operates on the depth-map function f. One needs a priori knowledge of all surface reflectances and all illumination sources of the scene to invert this I operator. Since range-image formation is conceptually a simpler process than intensity-image formation, to recover information of the real object using range data is correspondingly easier.

However, one inherent problem that always exists even when range information is available is that the so-called 3-D range information is in fact only $2\frac{1}{2}$-D since different parts of an object become invisible when seen at different angles. Therefore all 3-D object recognition problems are inherently occlusion problems, even at the absence of any other objects.

Most existing techniques that handles 3-D object recognition utilizing 3-D information are based on abstracting the object by some high level features. Singularity points, edges, planes, quadratic surface patches, cylinders and cones are the usual primitives used in constructing a relational graph for representation and matching. Bhanu[23] presented an algorithm for 3-D scene analysis which is a generalization of their 2-D scheme [4].

Planar surfaces were used to approximate the object and a stochastic face-labeling scheme was used to establish a correspondence for each of the faces obtained from the scene with those from the model. However, as pointed out in [19], it relies too heavily on the consistency of the output from the face-finding algorithm. Ballard and Sabbah [24] developed a scheme which emphasized on explicitly decoupling orientation and translation parameters and employed techniques similar to Hough transform to estimate the parameters. Faugeras and Hebert [25] at INRIA have devised a 3-D object recognition algorithm based on geometrical matching between primitive surfaces. Oshima and Shirai [26] devised a scheme which designs for each view of the object a characteristic view consisting of a kernel region which is used for matching during recognition. Their scheme is interesting since it can handle more than one object at a time. However because of the view-dependent nature of the stored object models, matching becomes slow when many views are allowed. Horn [27] discussed the use of extended Gaussian image for object recognition. The method is ideal for convex object recognition without occlusion. There exists many other techniques in this subject. Again see [19] for a more detailed review.

The above mentioned two categories use either intensity-image or range-image exclusively, whereas there is a trend in this subject that multiple sensory data are used to infer a solution. Magee et al.[29] performed experiments in 3-D object recognition using intensity guided range sensing. Although their main purpose was to reduce the time required to acquire a com-

plete range map, the idea is valuable and can be argumented by the fact that human vision systems rely heavily on intensity images; we can interpret a black and white photograph precisely. Therefore we believe a recognition system using multiple sensory data should be more powerful and efficient compared to those using only either intensity image or range data.

As already mentioned, most existing techniques are based on abstraction of the object by certain high level primitives. Some assume the existence of vertical and horizontal planes, others directly deals with the extraction of cylindrical or conical surfaces. Yet some of them can handle polyhedras only, and others rely on the consistency of the polyhedral approximation for the same object at different views. In one way most techniques are not well-suited to the recognition of some truly irregular-shaped objects. Also the time required for recognition in previous schemes are usually too long to be practical in real-time applications.

The approach we adopted in this work is quite different from any existing approach. We are primarily working on range data. Instead of extracting some high-level features for matching, we directly perform low-level subtemplate matching. The model object is broken down into many subtemplates which are surface patches of the object's boundary surface. Each of these subtemplates is directly matched against the rotated object in the scene. The algorithm makes no assumption on the model object such as the existence of relatively planar surfaces, edges, singularity

points, quadratic surface patches, cylindrical or conical entities. Therefore the algorithm is well-adopted to the recognition of irregular, arbitrarily-shaped objects. On the contrary, the method may turn out to be quite dumb when working on regularly-shaped objects with symmetry such as cylinders and cubes. Anyway we cannot expect that a single technique can handle all different problems and we believe that a complete object recognition system can never rely on just one or even a small number of different techniques, but should consists of different techniques working in cooperation to handle different entities. The subtemplate matching here can still be viewed as feature extraction, just as planes or cylindrical surfaces are extracted in other schemes, but more precisely model-driven extraction of irregularly-shaped surface patches. With different feature extractors working together simultaneously and then apply rigidity constraints and contextual information, the resulting system should be able to work properly and efficiently in a general sense. Fig.3.1 illustrates the structure of such a system.

The algorithm we propose here is in principle very similar to the one we introduced in chapter 2. Subtemplate matching is done in the first stage and dynamic programming is again used in the second stage to produce a consistent interpretation of the scene under study. Details of the algorithm will be given in the following sections.

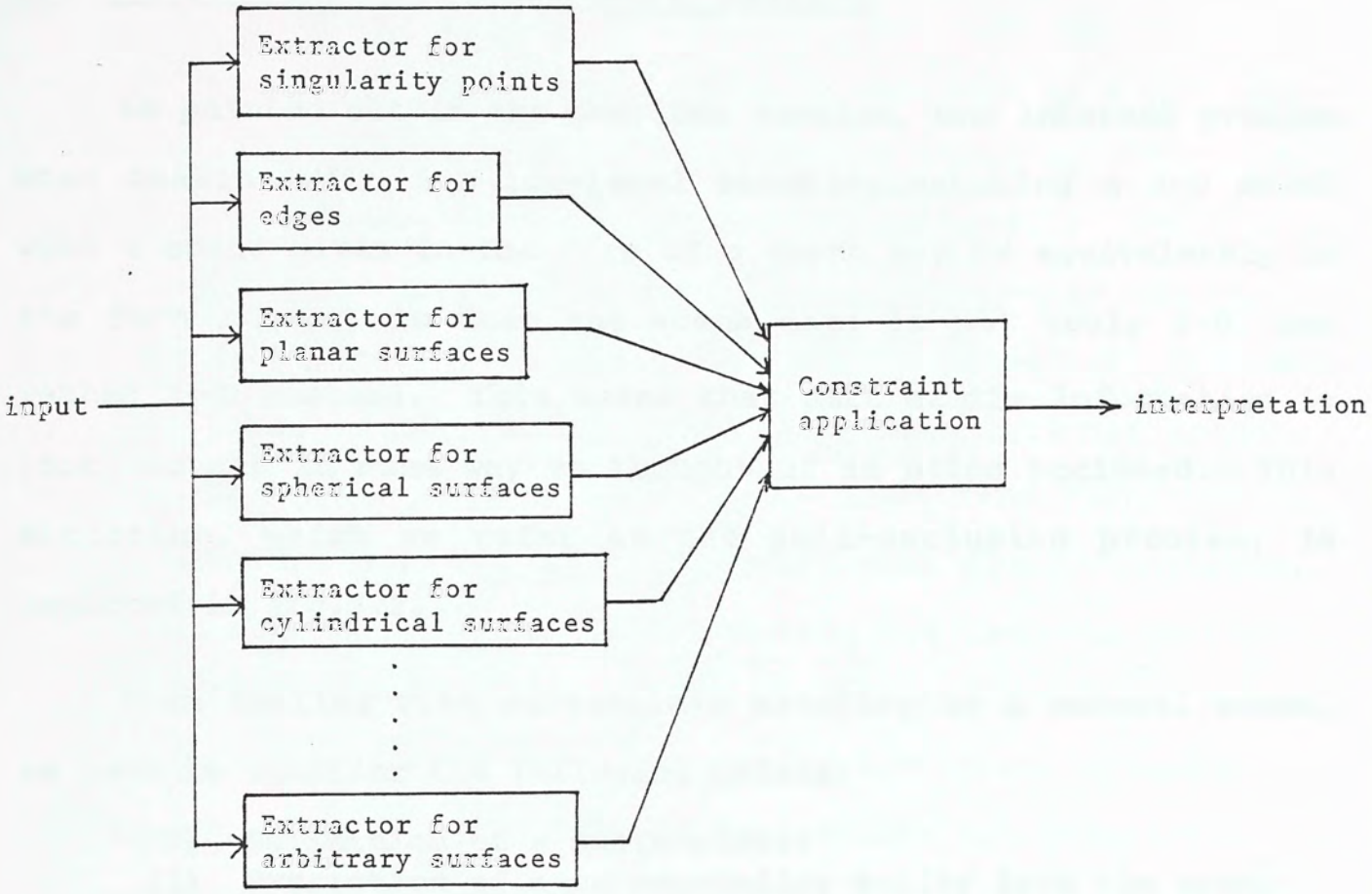


Fig 3.1 A 3-D object recognition system with non-homogeneous feature extractor.

3.2 Three-Dimensional Subtemplate Matching

As pointed out in the previous section, one inherent problem when dealing with 3-D low-level matching(matching a 3-D model with a scene given in the form of a depth map or equivalently in the form $z=f(x,y)$) is that the scene data is not truly 3-D, but rather $2\frac{1}{2}$-D instead. This means that part of the information is lost, or can in some way be thought of as being occluded. This situation, which we refer as the self-occlusion problem, is depicted in fig.3.2.1.

When dealing with subtemplate matching in a general sense, we have to consider the following points:

i) Definition of a subtemplate;
ii) Extraction of a corresponding entity from the scene;
iii) Alignment of the two entities;
iv) Actual matching.

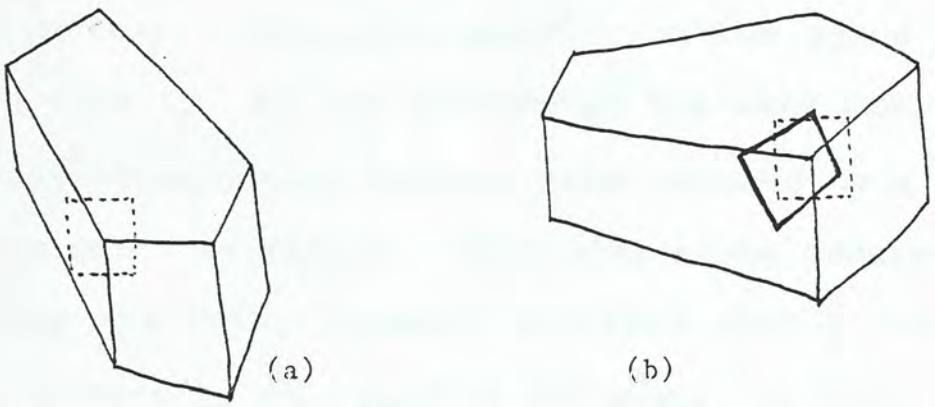
For the 2-D case the answers are straightforward. A subtemplate is nothing but a segment of the model template boundary of length ℓ headed by a reference point ζ_B. When a point ζ on the object boundary curve is suspected to match with ζ_B, an equal length segment headed by ζ is extracted for matching. Alignment is done by calculating the average angle of slope for both segments, and rotate the subtemplate by that difference around ζ_B. Matching is done by translating ζ_B to ζ and calculate and sum the distances between each corresponding point on the two equal length segments.

For the 3-D case the situation is not as simple, mainly due to the additional freedom of rotation and the self-occlusion problem. Assume that a model is given in the form of a collection of depth-maps corresponding to different views of the model. First of all we have to define what a subtemplate is. Note that simply defining the subtemplate to be a rectangular sub-map of the depth map and similarly extracting a same-sized rectangle from the scene would not work since the subtemplate has to undergo an arbitrary rotation in 3-space. After the rotation the rectangular frame is distorted. Ideally this distorted rectangle should be aligned with the object surface and then the projection is extracted and used for matching. However the alignment depends on the estimation of the orientation for both entities for matching. In other words we have to estimate the orientation of the patch so that we can align and project the subtemplate on the object surface to extract the patch for matching, but the estimation of the orientation in turns depends on the patch. This situation is depicted in fig.3.2.2. This contradictory condition can be resolved if we use a 3-D windowing scheme which is invariant to 3-D rotation instead of using a 2-D window such as a rectangular window or a circular window. The main point is that the outcome of the scheme when applied to the same portion of the surface but viewed at different angles should remain the same. The only window satisfying this criteria is a spherical window. This situation is depicted in fig.3.2.3.

Model
(3-D data)

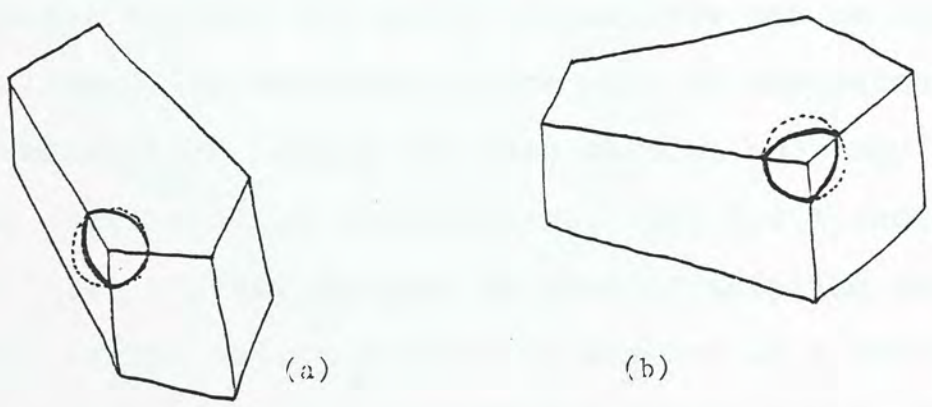
Scene
(2½-D)

Fig 3.2.1 3-D vs 2½-D data



(a) (b)

Fig 3.2.2 Defining a subtemplate using 2-D rectangular window. The extracted portions(bounded by dotted lines) for the same object at different orientations are not identical. The distorted rectangle(heavy line in (b)) corresponds to the correct portion that should be extracted for matching with that in (a).



(a) (b)

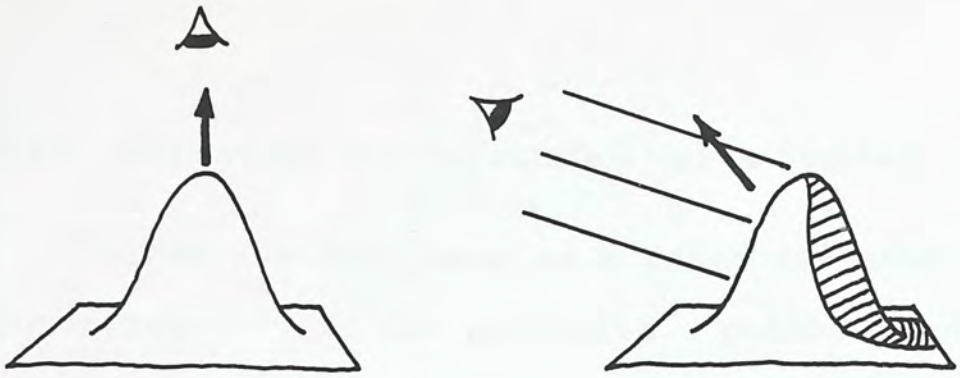
Fig 3.2.3 Defining a subtemplate via a 3-D spherical window. The extracted portions (bounded by heavy lines which are intersections of the spheres with the object surfaces) are identical for both views.

3.2.1 Definition of a subtemplate & extraction of an object patch

Given a model in the form of a set of 3-D discrete points(or preferably a collection of depth-maps corresponding to different views of the model), a subtemplate is specified by a centre point ζ_g on the surface of the model, and a radius r which defines the size of the subtemplate. All the surface points included within the sphere with ζ_g as centre and r as the radius constitute the subtemplate for matching. Similarly given a scene again in the form $z=f(x,y)$. Suppose a point ζ on the scene is suspected to match with ζ_g. We can go through the same process to extract a corresponding object surface patch bounded by a sphere with ζ as centre and r as radius. This step alone ensures that identical patches are being matched, provided that ζ does match with ζ_g. However this is only part of the story. We still have to face the self-occlusion problem. The object patch extracted for matching may not be complete; some part of it may be invisible.

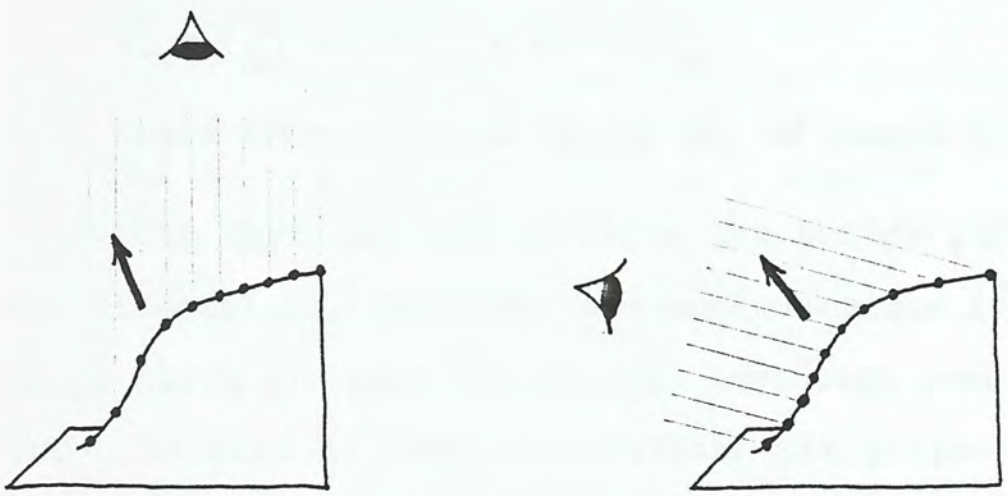
Once the subtemplate as well as the object patch for matching is defined, the orientation of both patches must be estimated so that the model subtemplate can be properly oriented and aligned for matching. Since part of the patch may be missing, the estimation using the seen portion only may turn out to be quite erroneous or inconsistent. Fig.3.2.4 shows such a situation. Yet another problem is that of sampling density. Since the depth of the object surface is sampled in a uniform rectangular grid, the resulting sampling density will be different if the

surface is viewed at different angles. Therefore unless some sort of re-sampling that emulates true uniform surface sampling is done, the estimated orientation will be biased. This is shown in fig.3.2.5. However such a re-sampling scheme is by no means simple. Our method here provides a useful guideline for estimating the orientation to a reasonable accuracy by using only part of the patch, namely the outer boundary of the patch only.



(a) (b)

Fig 3.2.4 Error in orientation estimation due to self-occlusion. The estimated orientation(the heavy arrows) for the same surface patch at different viewing angles turns out to be inconsistent due to the self-occlusion problem. The shaded region in (b) corresponds to the invisible portion.



(a) (b)

Fig 3.2.5 Error in orientation estimation due to non-uniform surface sampling. Variation of sampling density along the object surface results in the inconsistent orientation estimation for the same patch viewed at different angles.

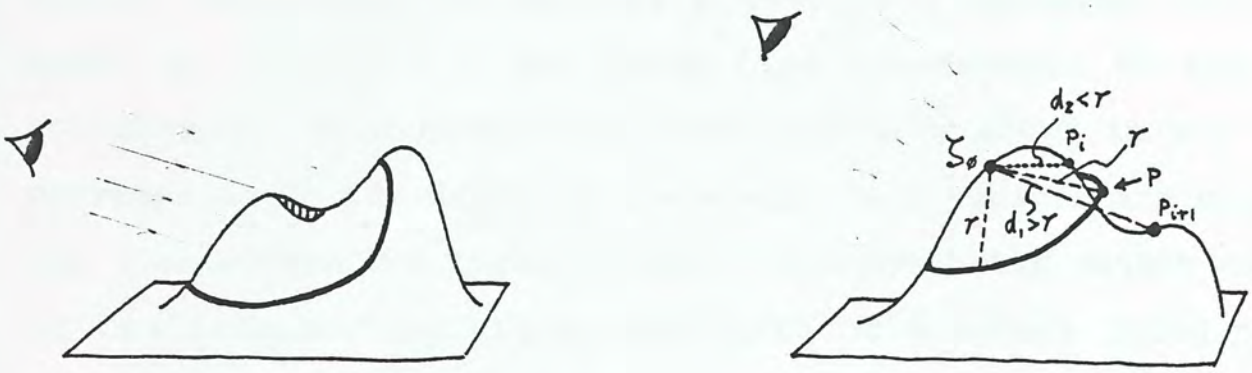
3.2.2 Estimating the orientation of a surface patch

Given a subtemplate or a patch for matching, points on the outer boundary of the patch(i.e. points exactly r away from the centre point ζ_0) forms one or more closed space curves $a(s)$. For simplicity we assume there is only one such $a(s)$. This can be realistic if r is properly chosen. However the algorithm can be easily modified to handle more than one such space curves. We may define a vector $\vec{v}_a$ which is associated with the surface patch defined by $a(s)$ as follows: (see fig.3.2.6c)

$$\vec{v}_a = \frac{1}{n}\sum_{i=0}^{n-1}\left[\overrightarrow{(a(i+1)-\zeta_0)} \times \overrightarrow{(a(i)-\zeta_0)}\right] \qquad (3.1)$$

where $a(n)=a(0)$ and n is the no. of sample points of $a(s)$.

Note that instead of using the entire patch in estimating the orientation, the outer boundary $a(s)$ alone is used. Using the space curve $a(s)$ only has several advantage over using the entire patch in that i) less calculation; ii) proper selection of the subtemplate position and size can to a certain extent minimize the effect of self-occlusion, as illustrated in fig.3.2.6a; iii) even when part of the space curve is invisible in the scene, a reasonable estimation can still be obtained by the algorithm proposed in Appendix A for obtaining the space curve which uses interpolation to obtain an approximation for the invisible part, as illustrated in fig.3.2.6b; iv) uniform surface re-sampling is difficult but it is easy to resample the space curve such that all points are equi-distant.



(a) (b)

Fig 3.2.6 Using only a(s) to minimize the effect of self-occlusion

(a) Though part of the patch is invisible(the heavily-shaded part), the outer boundary a(s) (in heavy line) is completely visible.

(b) Even when part of a(s) is invisible, a point p is obtained which is r away from the centre point by interpolating between two consecutive visible points p_i & p_{i+1}, the former being too close to ζ_ϕ while the latter too far away. The obtained a(s) passes through p and approximates the true a(s).

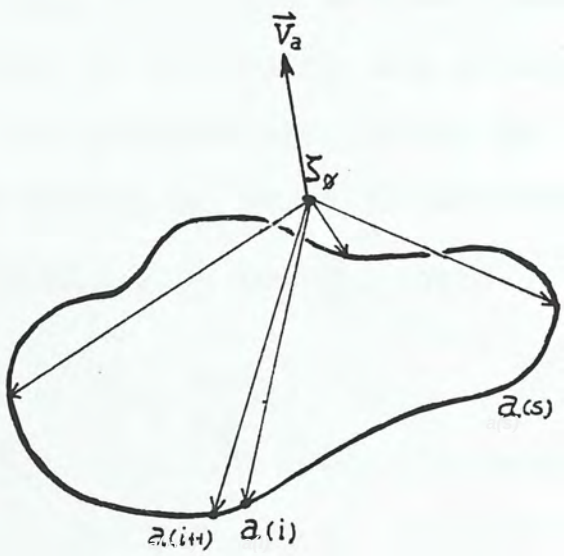


Fig 3.2.6c Definition of $\vec{V}_a$ for a space curve a(s).

3.2.3 Performance in orientation estimation

In order to see the effectiveness of such a scheme in orientation estimation, we perform a test on a synthetic object as shown in fig.3.2.7. The heavy line corresponds to the outer boundary of the subtemplate. The intensity shown in the figure corresponds to the depth of the scene; dark regions are closer to the viewer than the light regions. The synthetic object consists of two intersecting planes with part of a sphere lying between them. The orientation of a specific patch defined by ζ_β and r is estimated using various schemes. Then the object is rotated in three space to generate rotated versions of the same depth map. Two of them are shown in fig.3.2.8. After that the orientation of the corresponding patch centered at the rotated $\zeta_\beta(x_\beta, y_\beta)$ is estimated using various schemes.

Fig.3.2.9 shows the result when the entire patch is used for the estimation. No resampling of any sort is performed and no special care is taken to handle self-occlusion. The orientation is simply estimated by averaging the individual orientation at every point j of the subtemplate, which is in turns estimated by using 6 neighbour points in the x direction and another 6 in the y direction to calculate $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$, where

$$\left.\frac{\partial z}{\partial x}\right|_{x_0,y_0} = \frac{1}{6} \sum_{\substack{i=x_0-3 \\ i\neq x_0}}^{x_0+3} \frac{[z(i,y_0) - z(x_0,y_0)]}{(i - x_0)},$$

$$\left.\frac{\partial z}{\partial y}\right|_{x_0,y_0} = \frac{1}{6} \sum_{\substack{i=y_0-3 \\ i\neq y_0}}^{y_0+3} \frac{[z(x_0,i) - z(x_0,y_0)]}{(i - y_0)} \qquad (3.2)$$

The orientation at that specific point j is then given by $\vec{v}_j = (\frac{\partial z}{\partial x}, \frac{\partial z}{\partial y}, 1)$. The orientation of the entire patch centered at ζ_0 is calculated as

$$\vec{V} = \frac{1}{k} \sum^{k} \vec{v}_j \ , \qquad (3.3)$$

where k is the number of points seen in the patch.

Since the amount of rotation is known, we can pre-compute the orientation vector for the rotated patch and compare it with the estimated one. The error is defined as the agnle between the exact orientation vector and the estimated orientation vector.

From the figure it can be seen that the estimation is very poor except at very small degree of rotation. Beyond that the error rises in a linear fashion. The error here comes mainly from the inconsistent sampling density when the scene is rotated. In fact the effect of self-occlusion hardly shows up. One may use other schemes in estimating the orientation at each specific point, say by fitting a plane to a 3 by 3 window centered at that point. The result may turn out to be better.

Fig.3.2.10 shows the result of our scheme, ie. using only the outer boundary in the estimation. From the figure it can be seen that the result is quite good, especially at small rotation. For rotation smaller than 50° the error in estimation is quite random. Beyond that the self-occlusion problem shows its effect and the error gradually increases. However the error is still relatively small, less than 3° at a rotation of 70°. From that we can be quite confident with our scheme in estimating orientation.

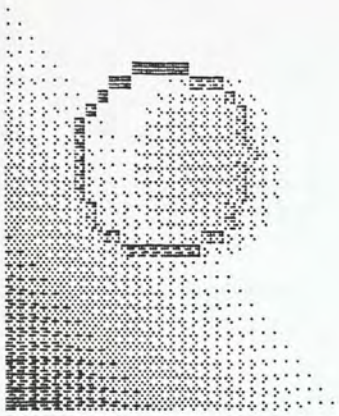
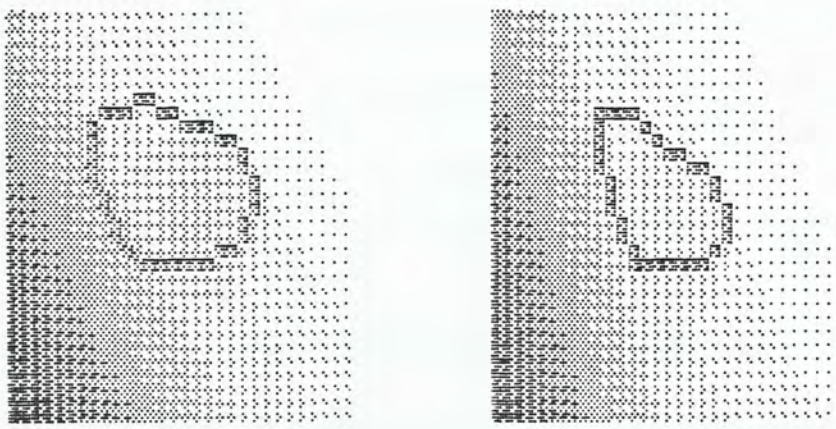


Fig 3.2.7 A synthetic depth map
with subtemplate shown.



(a) (b)

Fig 3.2.8 Rotated views of fig.3.2.7.
(a) 34°; (b) 52°

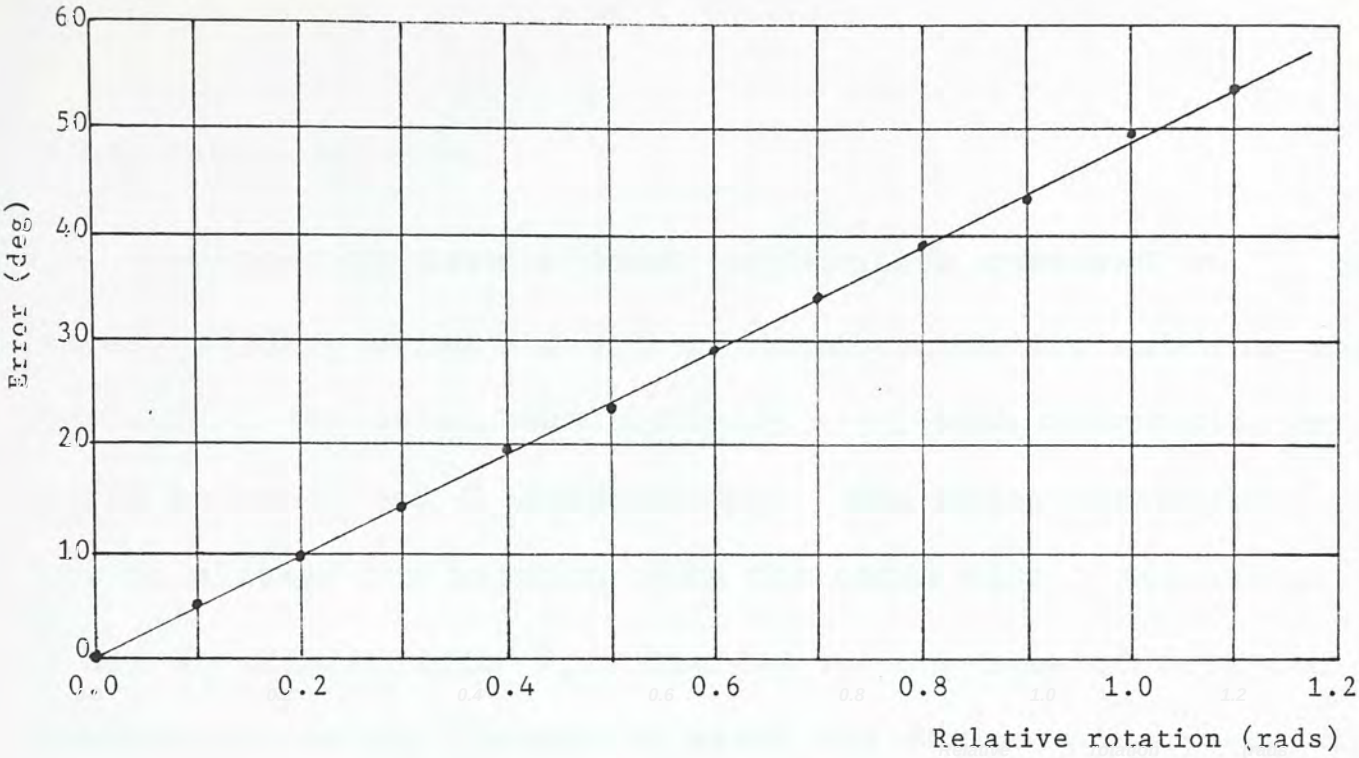


Fig 3.2.9 Error in orientation estimation using the entire patch

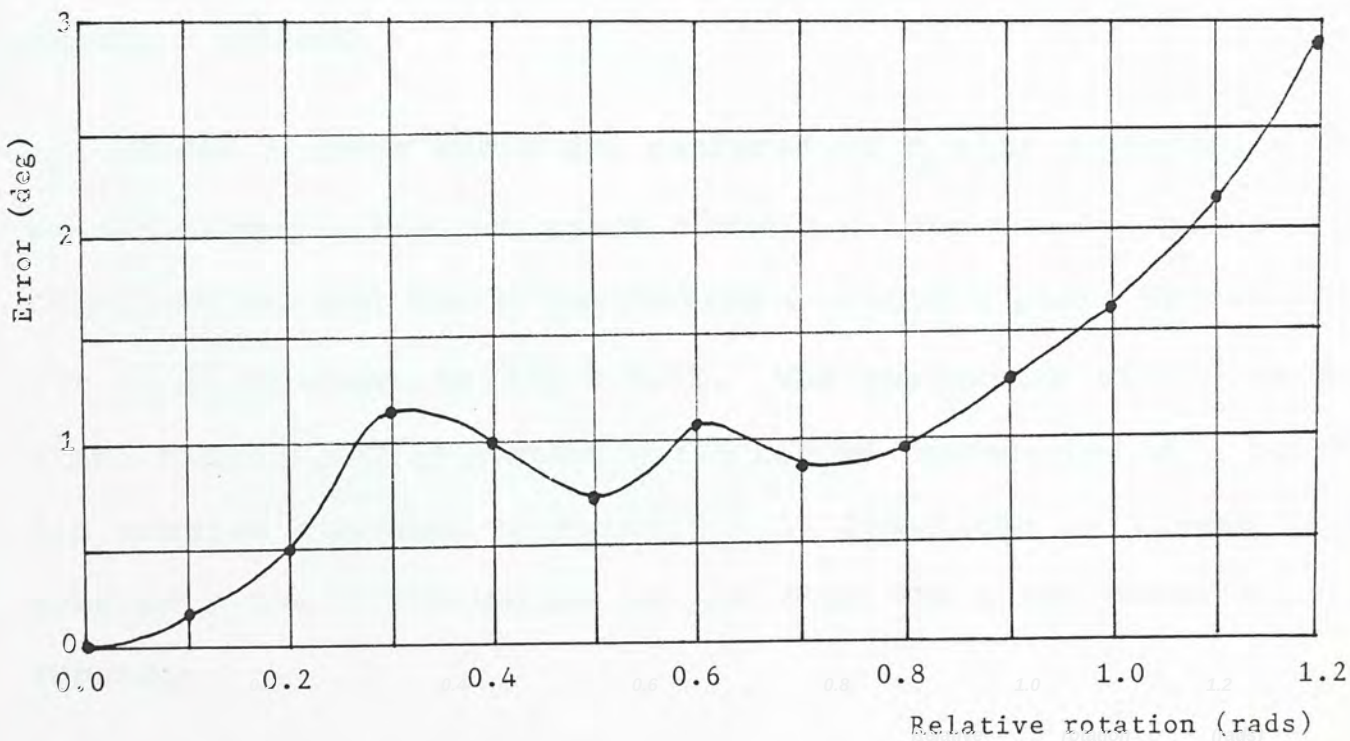


Fig 3.2.10 Error in orientation estimation using only the outer boundary

3.2.4 Actual matching

Now that we have a model subtemplate centered at ζ_g, the outer boundary being $a(s)$ and an object patch for matching centered at ζ, the outer boundary being $a'(s)$, with orientation estimated to be $\vec{v}_a$ and $\vec{v}_{a'}$ respectively. The model subtemplate can now be aligned for matching with the patch with ζ_g translated to ζ and $\vec{v}_a$ aligned with $\vec{v}_{a'}$. Similar to the case of orientation estimation, we may attempt to match the entire subtemplate with the patch. Since there is still one rotational freedom for the model patch (rotation about ζ_g), the computation may be quite labourious. One must rotate the subtemplate about ζ_g and at each different rotation calculate for each point on the subtemplate its distance from the object patch. On the contrary if we attempt to match $a(s)$ with $a'(s)$ first, the involved complexity can be somewhat reduced.

Given a space curve $a(s)$ centered at ζ_g with orientation $\vec{v}_a$, we can decouple the 3-D space curve $a(s)$ into two 1-D parametric functions $c(s)$ and $z(s)$ by projecting $a(s)$ onto a plane perpendicular to $\vec{v}_a$ as shown in fig.3.2.11. The projection of $a(s)$ on the plane forms a closed contour which can be represented by a periodic complex function $c(s)=x(s)+jy(s)$ if the plane is viewed as a complex plane. The height of $a(s)$ over the plane forms a real function $z(s)$.

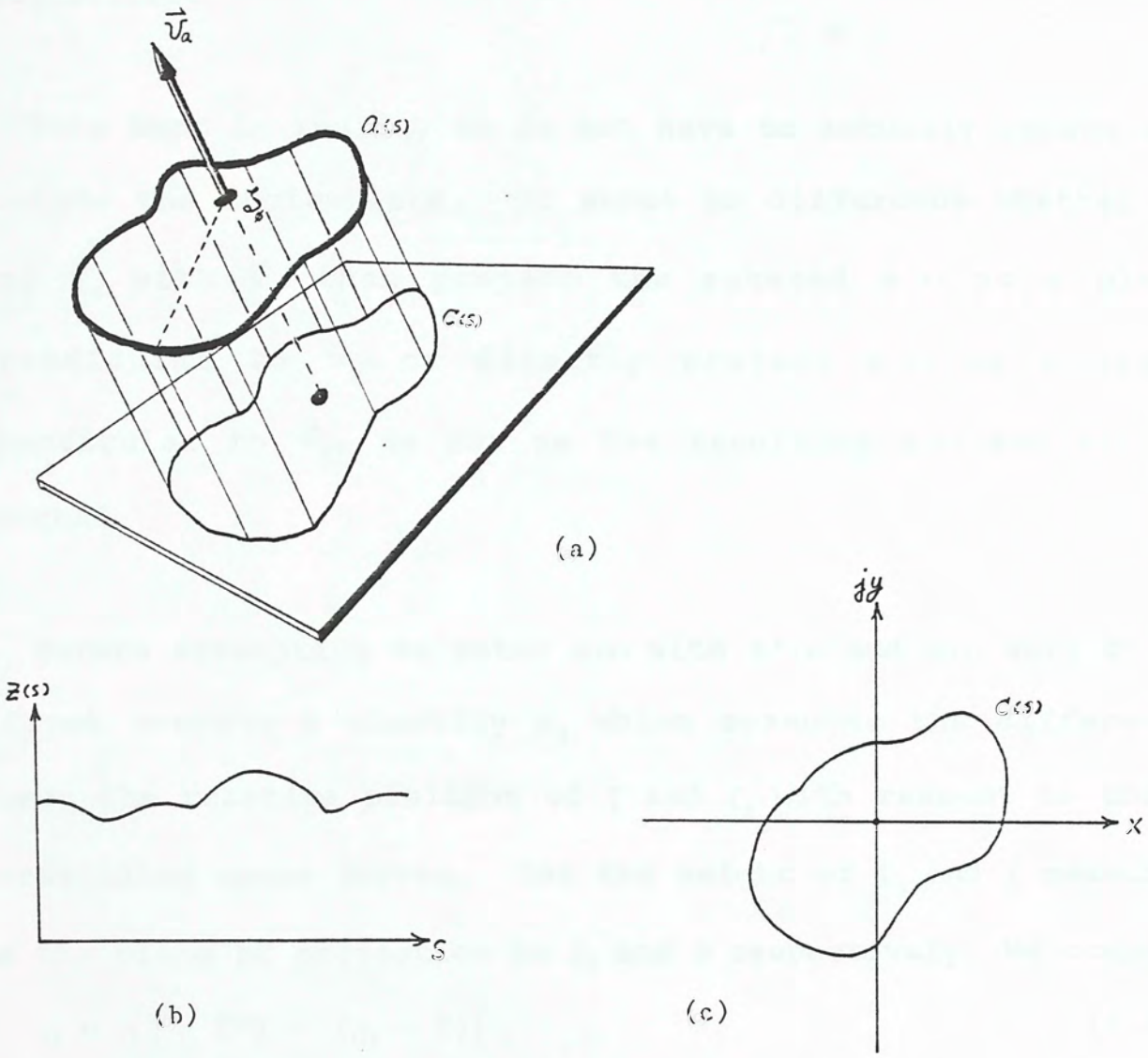


Fig 3.2.11 Decoupling of $a(s)$ into (b) $z(s)$ and (c) $c(s)$

In this way we decouple the space curve of the object patch $a'^{(s)}$ into $\{z'^{(s)}, c'^{(s)}\}$. We then translate and rotate the space curve of the model subtemplate $a^{(s)}$ such that ζ_β aligns with ζ and $\vec{v}_a$ aligns with $\vec{v}_{a'}$. We then similarly decouple the transformed $a^{(s)}$ into $\{z^{(s)}, c^{(s)}\}$ and match them against $\{z'^{(s)}, c'^{(s)}\}$. This is shown in fig.3.2.12.

Note that in reality we do not have to actually rotate and translate the subtemplate. It makes no difference whether we align $\vec{v}_a$ with $\vec{v}_{a'}$ then project the rotated $a^{(s)}$ on a plane perpendicular to $\vec{v}_{a'}$ or directly project $a^{(s)}$ on a plane perpendicular to $\vec{v}_a$, as far as the resulting $z^{(s)}$ and $c^{(s)}$ is concerned.

Before attempting to match $z^{(s)}$ with $z'^{(s)}$ and $c^{(s)}$ with $c'^{(s)}$, we first compute a quantity e_β which measures the difference between the relative position of ζ and ζ_β with respect to their corresponding space curves. Let the height of ζ_β and ζ measured from the plane of projection be h_β and h respectively. We compute

$$e_\beta = \left|(h - \bar{z}') - (h_\beta - \bar{z})\right| , \tag{3.4}$$

where $\bar{z} = \frac{1}{n}\sum_{s=\beta}^{n-1} z^{(s)}$ and $\bar{z}' = \frac{1}{n}\sum_{s=\beta}^{n-1} z'^{(s)}$, (3.5)

as illustrated in fig.3.2.13.

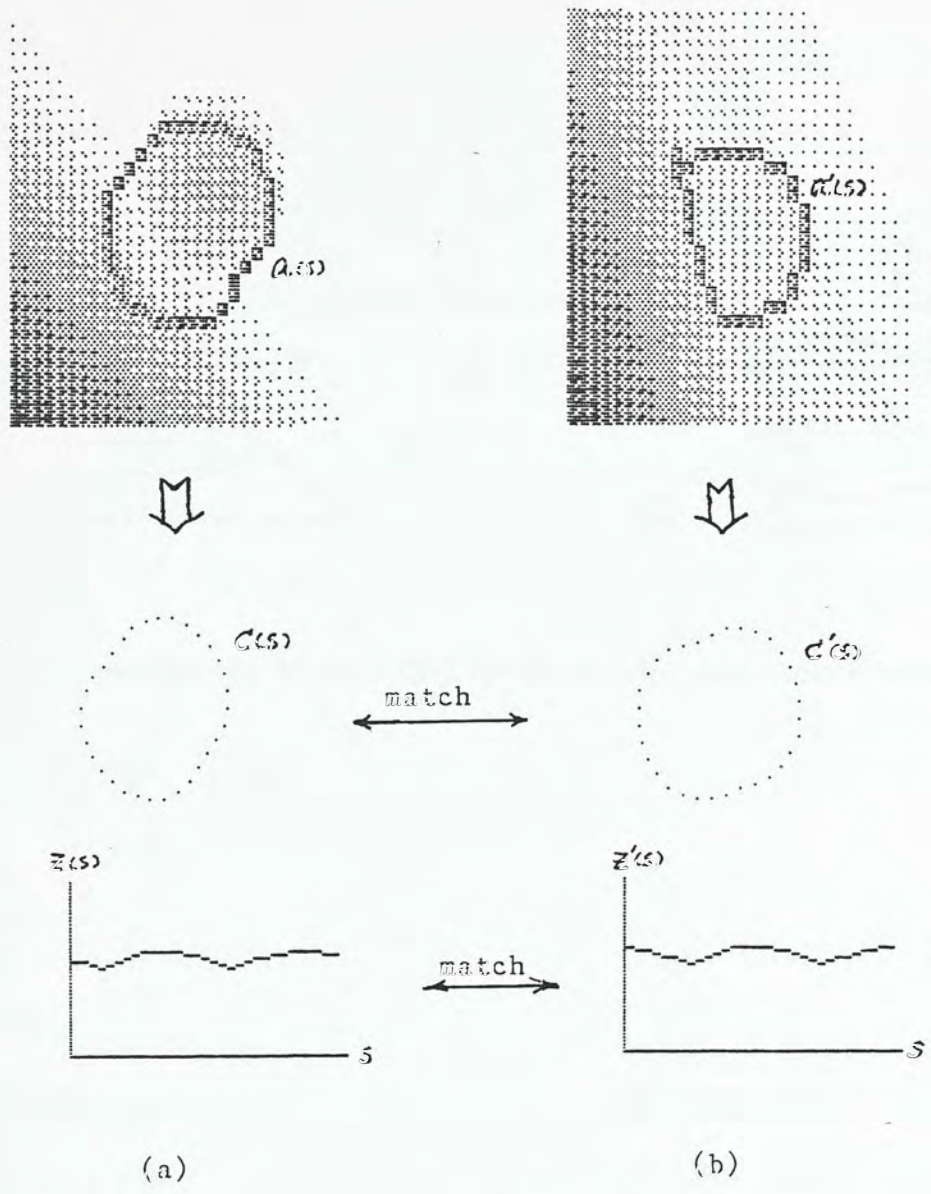


Fig.3.2.12 Decoupling of the space curve.
(a) model patch; (b) object patch.

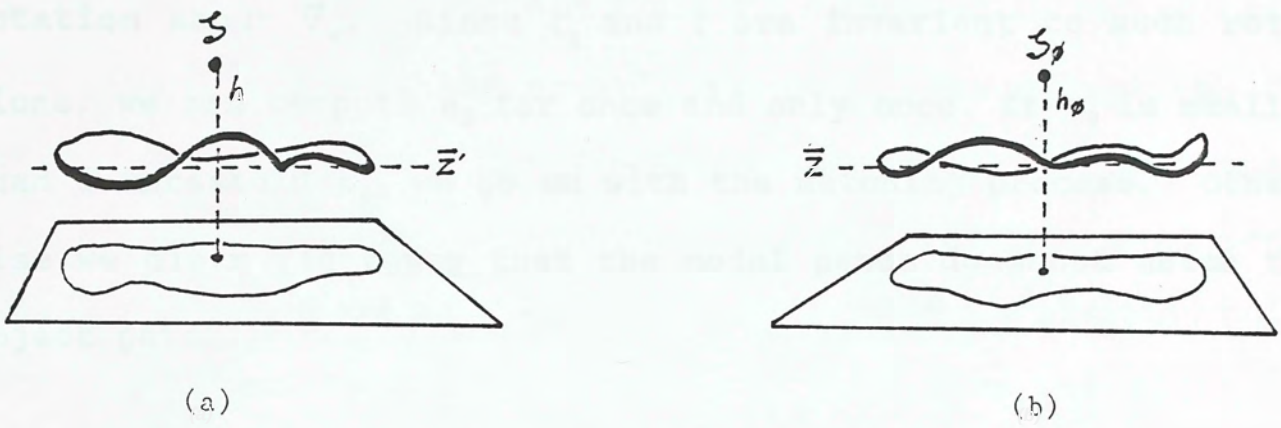


Fig 3.2.13 Computation of e_o. (a) model patch; (b) object patch.

Note that if we attempt to match the entire patch, all other points should be processed in the same way for each different rotation about $\vec{v}_{z'}$. Since r_θ and r are invariant to such rotations, we can compute e_θ for once and only once. If e_θ is smaller than a threshold t_θ, we go on with the matching process. Otherwise we claim rightaway that the model patch does not match the object patch.

We then match $z(s)$ against $z'(s)$ by computing

$$e_1 = 1 - \frac{\underset{\tau}{\text{Max}} \sum_{s=\emptyset}^{n-1} [z(s-\tau)-\bar{z}][z'(s)-\bar{z}']}{\left[\sum^{n} |z(s)-\bar{z}|\right]\left[\sum^{n} |z'(s)-\bar{z}'|\right]} \qquad (3.6)$$

That is, we compute the cross-correlation of $z(s)$ and $z'(s)$ and take the maximum, normalized and subtracted from 1 to measure the error in matching $z(s)$ and $z'(s)$. τ corresponds to the shift in starting point to get the best match. If e_1 is smaller than a threshold t_1 we can go on matching $c(s)$ with $c'(s)$ by shifting $c(s)$ by τ. To be more cautious one may record more than one such τ since the best τ in matching $z(s)$ with $z'(s)$ may not necessarily correspond to the best starting point shift in matching $c(s)$ with $c'(s)$ due to noise. However in our implementation, we retain only the best match just for simplicity.

Next we compute e_2 which reflects the error in matching $c(s)$ with $c'(s)$. Fourier Descriptor method is used here to estimate the required rotation for $c(s)$ about $\vec{v}_a$, such that it matches $c'(s)$ best. Other methods may as well be used. For example we can convert $c(s)$ and $c'(s)$ into θ-s representations and than apply similar techniques as in section 2.2. FD method is used here, taking the advantage of the efficiency of FFT. The starting point shift τ deduced in computing e_1 can be used here, or it can be cross-checked with that obtained independently using the FD method similar to that proposed by Wallace and Wintz [20]. In the former case, the computation is much simplified.

Since $c(s)$ and $c'(s)$ are periodic functions of s, they can be represented by Fourier series expansions

$$c(s) \Leftrightarrow C(w) \quad \text{and} \quad c'(s) \Leftrightarrow C'(w) \tag{3.7}$$

Rotating $c(s)$ by the angle θ in the complex plane requires multiplying each coordinate by $e^{j\theta}$. By linearity, multiplying each component of $C(w)$ by $e^{j\theta}$ is the equivalent frequency domain operation. If $c(s)$ and $c'(s)$ are obtained by tracing in the counter-clockwise direction, $C(1)$ and $C'(1)$ will always be the largest coefficients.

We proceed by rotating $C_{(s)}$ such that the phase of $C_{(1)}$ and $C'_{(1)}$ are the same. That is, we multiply each component of $C_{(w)}$ by $e^{j\theta_T}$, where

$$\theta_T = \angle C'_{(1)} - \angle C_{(1)} \tag{3.8}$$

Thus we compute e_2 by

$$e_2 = \sum_{w=0}^{n-1} |C_{(w)}e^{j\theta_T} - C'_{(w)}| \tag{3.9}$$

Finally the overall matching error e is computed by

$$e = k_0 e_0 + k_1 e_1 + k_2 e_2 \tag{3.10}$$

where k_0, k_1 and k_2 are constants for adjusting the relative weights of individual errors. In our experiment they are all set to 1.

If e is smaller than a threshold t, ζ is accepted to be a probable match for ζ_0. Recall that in the 2-D case, a reference point is defined for each object. For the three-dimensional case, since two vectors uniquely determines the position and orientation of an object in 3-space, we arbitrarily select three reference points for each object. As the matching process goes by, the subtemplate is rotated and aligned on the object surface. The reference points are correspondingly rotated in three space. Their final positions correspond to the accumulators to be incremented if Hough Transform is implemented. As in the 2-D case, we instead explicitly set up a match table to store the potential good matches. The matching position ζ, the estimated orientation at that point $\vec{v}_a$, and the positions of the reference points as

well as the matching errors are stored in the table. Again we restrict the size of the table to q by m, where q is the total number of subtemplates and m is the maximum number of good matches for each subtemplate.

We may now summarize the entire subtemplate matching process. Given a model in the form of a collection of depth maps at different views, we first construct a library of subtemplates by specifying for each subtemplate a centre point ζ_g and a radius r. For each such subtemplate we

1) obtain its outer boundary space curve $a(s)$,
2) estimate its orientation $\vec{v}_a$
3) project $a(s)$ onto a plane perpendicular to $\vec{v}_a$ to obtain $z(s)$ and $c(s)$,
4) calculate the Fourier transform of $z(s)$ and $c(s)$.

Note that all the above calculations can be computed offline with the results stored in the library. The Fourier transform of $z(s)$ is also computed since in reality the cross-correlation in calculating e_1 is also done in the frequency domain.

Now given an object scene again in the form of a depth map, we perform the following steps for each suspected point ζ(in our implemention all points within the depth map are taken to be suspected points).

5) Obtain the outer boundary space curve $a'(s)$,
6) estimate the orientation $\vec{v}_{a'}$
7) decouple $a'(s)$ into $z'(s)$ and $c'(s)$,
8) calculate the Fourier transform of $z'(s)$ and $c'(s)$,
9) compute e_g which measures the difference between the relative position of ζ and ζ_g with respect to their corresponding space curves,
10) if e_g is greater than t_g, we try the next subtemplate. Otherwise we proceed to

11) compute e_1 in matching $z_{(s)}$ with $z'_{(s)}$ using correlation,
12) if e_1 is greater than t_1, we try the next subtemplate. Otherwise we proceed to
13) compute e_2 in matching $c_{(s)}$ with $c'_{(s)}$ using FD method,
14) compute the overall matching error e. If e is smaller than t than ζ is accepted as a good match for ζ_β for the subtemplate under consideration. Enter the corresponding items into the match table.

One great advantage about the above scheme is that i)many entities can be calculated offline; ii)the matching is done in a hierarchical manner. e_β is most easy to compute, next comes e_1 and finally e_2. Obvious mismatches are rejected in early stage of the algorithm and thus the speed of the entire matching process is greatly enhanced.

Fig.3.2.15 shows the various matching errors e_β, e_1, e_2 and e in the form of an intensity map when the scheme is applied on the synthetic object shown in fig.3.2.14a. The centre of the subtemplate is at (17,17) and r=8. Fig.3.2.14b and c shows a rotated version of the same object. The correct matching point is known to be (15,18). The space curve obtained at that point with its various projections are shown in fig.3.2.14c, with those corresponding to an incorrect matching point shown in fig.3.2.14b. From fig.3.2.15e it can be seen that the best matching occurs at (15,18), which is shown as the brightest spot. This is in agreement with our prediction. Fig.3.2.16 shows the form of a match table with 3 subtemplates.

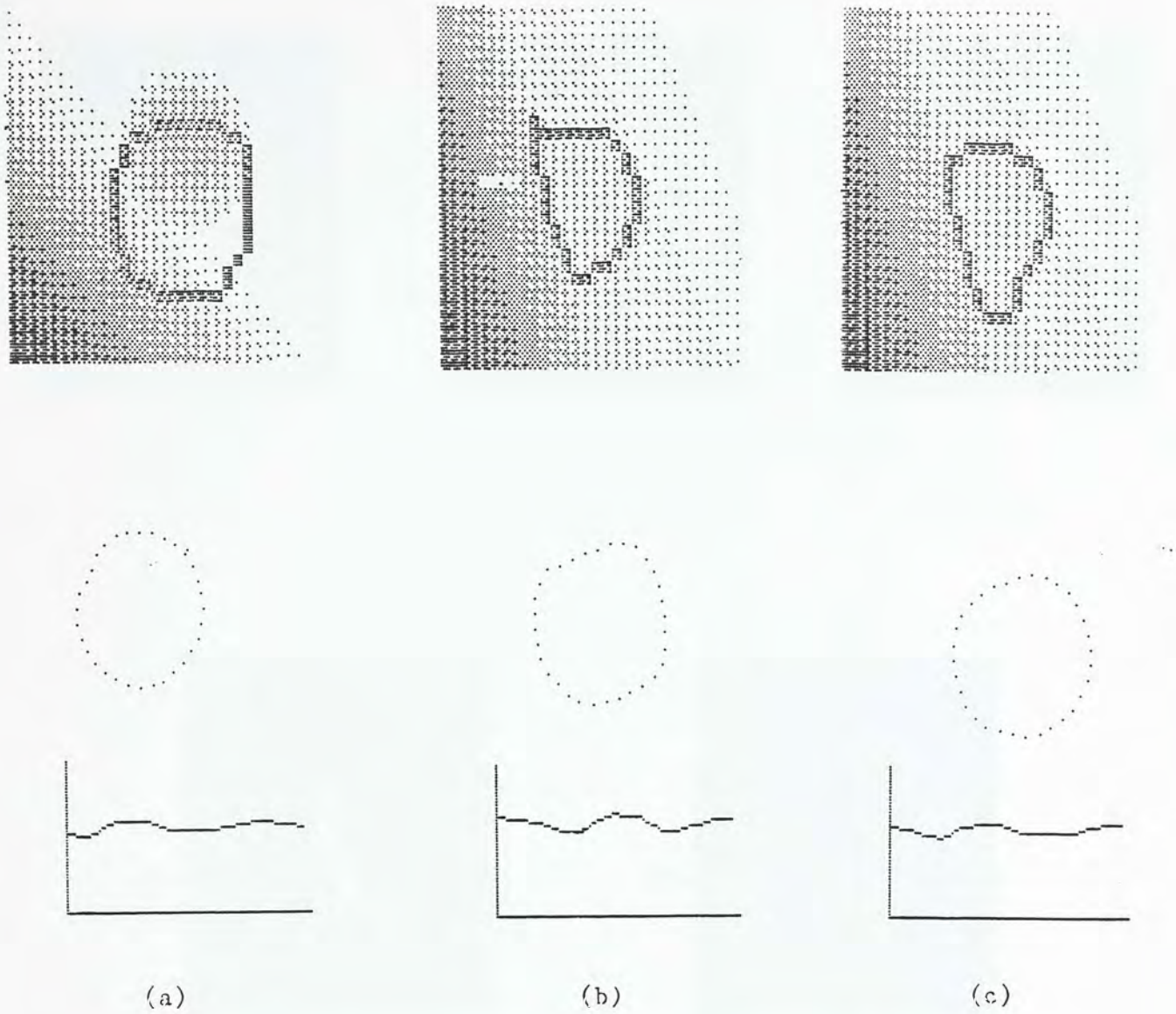
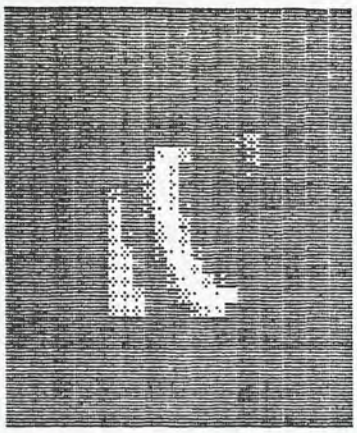
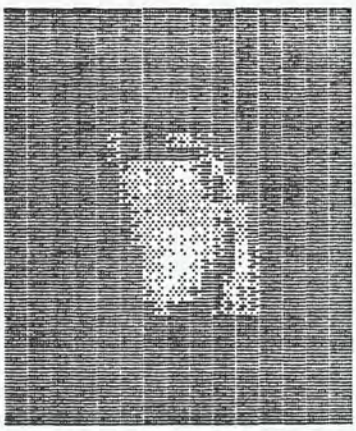


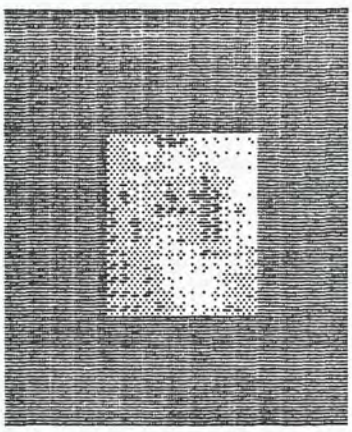
Fig 3.2.14 An example illustrating the actual matching.
(a) subtemplate and its projections.
(b) object patch centered at an arbitrary point.
(c) object patch centered at the best matching point.



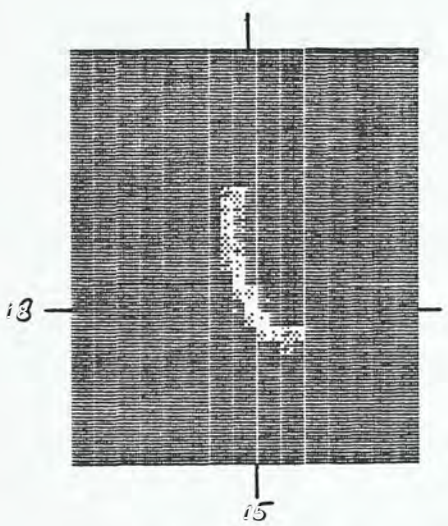
(a) e_0



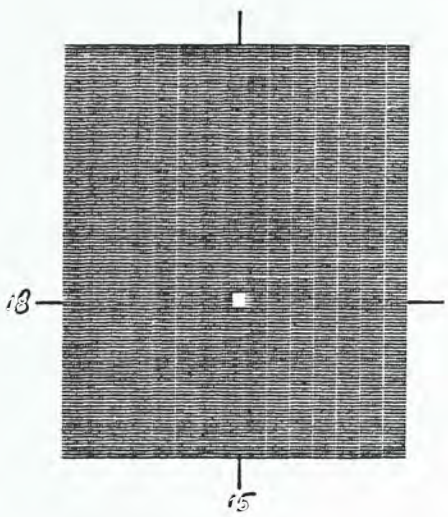
(b) e_1



(c) e_2



(d) Total error e



(e) e threshold to show the best matching position

Fig 3.2.15 Various matching errors in the experiment shown in fig 3.2.14

subtemplate	match no.	error	rotation	position	orientation	references					
0	0	0.1150	3.443	16,17,29.76	-0.757, 0.151,-0.635	0.85, 13.27, 20.59	-8.47, 18.76, 10.21	2.39, 25.28, 3.91			
	1	0.1298	-0.540	15,11,27.72	-0.630, 0.537,-0.561	3.93, 19.85, 21.04	-5.34, 24.75, 10.31	-8.81, 11.40, 7.22			
	2	0.2615	-0.184	15,13,29.31	-0.669, 0.423,-0.612	3.40, 17.94, 19.94	-7.30, 20.56, 9.75	-12.84, 7.80, 12.30			
	3	0.2793	3.256	16,16,29.81	-0.740, 0.202,-0.642	1.11, 13.88, 20.43	-8.24, 20.41, 10.69	2.19, 28.94, 6.40			
	4	0.2795	-0.363	15,12,28.67	-0.537, 0.492,-0.593	3.87, 19.10, 20.40	-5.95, 23.00, 9.74	-10.68, 9.68, 9.24			
	5	0.2887	2.951	15,14,29.74	-0.698, 0.354,-0.622	1.72, 15.94, 20.53	-7.50, 24.70, 12.58	1.99, 35.16, 13.09			
1	0	0.0575	-0.704	15,18,29.86	-0.775, 0.132,-0.619	7.35, 18.86, 18.67	-2.27, 23.75, 8.25	-5.94, 10.40, 5.38			
	1	0.1208	-1.011	14,16,30.24	-0.745, 0.257,-0.615	7.84, 21.30, 18.96	-1.70, 28.49, 9.89	-3.27, 16.71, 2.22			
	2	0.1440	-0.547	16,19,29.18	-0.781, 0.085,-0.619	7.22, 17.85, 18.66	-2.32, 21.64, 7.72	-7.02, 8.31, 7.21			
	3	0.2193	3.457	13,12,28.83	-0.676, 0.488,-0.551	7.21, 25.13, 22.63	-5.09, 31.87, 17.32	2.99, 41.33, 10.60			
	4	0.2284	-0.987	14,15,30.14	-0.728, 0.312,-0.610	8.12, 22.07, 19.07	-1.03, 30.00, 10.21	-3.03, 18.60, 2.08			
	5	0.2594	3.278	13,11,27.92	-0.660, 0.528,-0.534	7.20, 25.40, 23.02	-5.07, 32.10, 17.59	2.79, 43.07, 13.36			
2	0	0.1567	-0.429	11,18,33.31	-0.859, 0.244,-0.450	5.47, 20.09, 14.99	-4.54, 22.86, 4.17	-9.03, 9.46, 4.90			
	1	0.1902	-0.767	10,15,33.66	-0.852, 0.320,-0.414	6.22, 24.60, 15.82	-3.37, 30.25, 5.77	-6.42, 17.18, 1.32			
	2	0.2374	-0.610	12,16,30.08	-0.744, 0.297,-0.599	8.82, 22.31, 14.43	1.46, 26.64, 2.10	-2.25, 13.23, -0.40			
	3	0.2429	-0.149	13,19,29.65	-0.787, 0.139,-0.601	6.70, 15.46, 14.96	-2.06, 15.01, 2.80	-8.18, 3.21, 7.65			
	4	0.2569	-0.708	10,14,31.93	-0.798, 0.377,-0.470	7.51, 24.84, 15.52	-0.96, 31.11, 4.85	-4.66, 18.26, 0.24			
	5	0.2599	-1.052	11,13,28.94	-0.703, 0.460,-0.543	10.15, 28.69, 17.45	3.76, 37.56, 7.18	3.27, 25.71, -1.88			

Fig 3.2.16 A match table with 3 subtemplates, q=3 and m=6.

3.3 Selecting the Best Set of Matches using Dynamic Programming

Now that given a match table such as one in fig.3.2.16, we use a DP scheme to select the best, consistent set of matches just as we did in the 2-D case. In fact the scheme is more or less the same as in section 2.3, except for the definition of the various metrics.

Given a match table consisting matches of the form M_{ij}: the jth matching for subtemplate i. Each entry (i,j) contains the following items:

X_{ij} : matching position - where that match occurs,
γ_{ij} : matching error - how well does it match,
U_{ij} : orientation - orientation of the patch centered at the point where the match occurs,
β_{ij} : references - reference points for the object if this match is accepted.

Similar to the 2-D case, we first define two metrics d_1 and d_2 between two matches $(M_{ij}, M_{k\ell})$ which measures their difference:

$$d_1(M_{ij}, M_{k\ell}) = |(V_i - V_k) - (U_{ij} - U_{k\ell})| \qquad (3.11)$$

$$d_2(M_{ij}, M_{k\ell}) = |(P_i - P_k) - (X_{ij} - X_{k\ell})| \qquad (3.12)$$

where V_i is the absolute orientation of subtemplate i,
P_i is the absolute position of subtemplate i.

In addition to d_1 which corresponds to a rotational metric and d_2 which corresponds to a positional one, we define yet another metric d_3 which explicitly try to minimize the deviation in the object reference points in the final selected set of matches.

d_3 is defined as

$$d_3 = \sum_{p=1}^{3} \left| \beta_{ij}^{p} - \beta_{k\ell}^{p} \right| \tag{3.13}$$

which states that the object reference points on the scene should concentrate in single points if all the matches are compatible. β^p denotes the p^{th} object reference point. In fact this metric can as well have been used in the 2-D case.

The compatibility c between two matches is thus defined as

$$c(M_{ij} , M_{k\ell}) = \frac{\alpha_1}{1 + d_1} + \frac{\alpha_2}{1 + d_2} + \frac{\alpha_3}{1 + d_3} \tag{3.14}$$

where the α_i's are constants used to adjust the relative weight of the factors.

Now given the set of all M_{ij}'s, i=1 to q where q is the number of subtemplates in the match table, we calculate

$$\begin{aligned}
&1. \quad f_1(x_2) = \max_{x_1} [c(M_{1x_1} , M_{2x_2})] \\
&2. \quad f_2(x_3) = \max_{x_2} [f_1(x_2) + c(M_{2x_2} , M_{3x_3})] \\
&\quad \vdots \qquad\qquad \vdots \\
&k. \quad f_k(x_{k+1}) = \max_{x_k} [f_{k-1}(x_k) + c(M_{kx_k} , M_{k+1,x_{k+1}})] \\
&\quad \vdots \qquad\qquad \vdots \\
&q\text{-}1. \quad f_{q-1}(x_q) = \max_{x_{q-1}} [f_{q-2}(x_{q-1}) + c(M_{q-1,x_{q-1}} , M_{qx_q})] \\
&q. \quad f_q = \max_{x_q} f_{q-1}(x_q)
\end{aligned} \tag{3.15}$$

where $f_{k-1}(x_k)$ represents the optimal compatibility for the matchings corresponding to subtemplates 1, 2, . . . k if subtemplate k is assumed to be matched with the object at x_k.

Again we have to consider the case when no correct match occurs for a particular subtemplate. The compatibility in this case is given by

$$c(M_{ij}, M_{k\ell}) = \begin{cases} c_1(1 - e^{-(\frac{X}{a_1}+\frac{U}{\pi})}), & j=nil \text{ or } \ell=nil \\ c_2 + (1 - c_2)e^{-(\frac{X}{a_2}+\frac{U}{\pi})}, & j=\ell=nil \end{cases} \quad (3.16)$$

where $X = |P_i - P_k|$ is the absolute distance between the two subtemplates; $U = \cos^{-1} \frac{\vec{V}_i \cdot \vec{V}_k}{|\vec{V}_i||\vec{V}_k|}$ is the angle between the orientation vectors of the two subtemplates. Introduction of this term accounts for self-occlusion. If the orientation of two subtemplates differs by 180°, then it is most likely that if one is seen, the other is invisible. Fig.3.3.1 shows the effect of U on the nil-class compatibilities.

Again here c_1, c_2, a_1 and a_2 are the key parameters which affect the outcome of the algorithm. A high c_1 and c_2 with low a_1 and high a_2 favour the nil matches. If c_1 and c_2 are set too high, it may cause a trivial solution with all subtemplates labeled nil. Too low a c_1 and c_2 with high a_1 and low a_2 may cause incompatible matches to appear in the final assignment. Our strategy here is to give a good initial value for c_1 and c_2 (0.7 and 0.6 in our experiments) and fix a_1 and a_2 (both equal 5 in our experiments). After the DP is applied, an evaluation is done on the resulting assignment. The average and standard deviation of the final object reference points for all subtemplates within the assignment are computed.

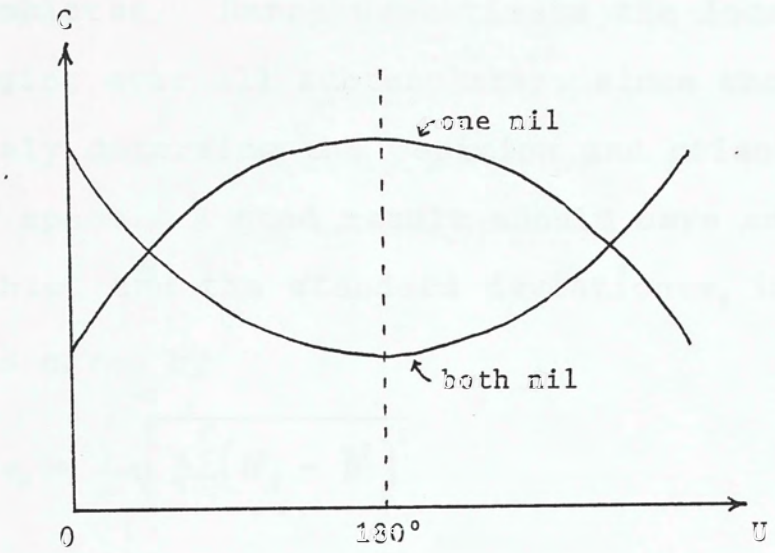


Fig 3.3.1 Effect of U on nil-class compatability.
U is the angle between the two orientation vector of the two subtemplates.

If the matchings are all accurate and correct the location of the three object reference points should be the same for all subtemplates. Hence we estimate the location of the object by averaging over all subtemplates, since three non-coplanar points uniquely determine the position and orientation of an object in three space. A good result should have as little nil-matches as possible, and the standard deviation σ_β in the object reference points given by

$$\sigma_\beta = \sum_{p=1}^{3} \sqrt{\frac{1}{q'} \sum_{t=1}^{q'} \left(\beta^p{}_t - \bar{\beta}^p \right)^2} \qquad (3.17)$$

should be small. $\beta^p{}_t$ denotes the p^{th} object reference point for the t^{th} subtemplate within the set of q' successfully labeled(non-nil) subtemplates and $\bar{\beta}^p$ denotes the mean of the p^{th} object reference point. We define a quantity η as a measure of the effectiveness of the solution as follows.

$$\eta = (1 - e^{-\frac{q'}{\alpha q}}) e^{-b\sigma_\beta} \qquad (3.18)$$

where q is the number of subtemplates used in the matching, q' is the number of non-nil matches in the final assignment. α, and b (typical values are 10 and 3 respectively) are adjustable parameters.

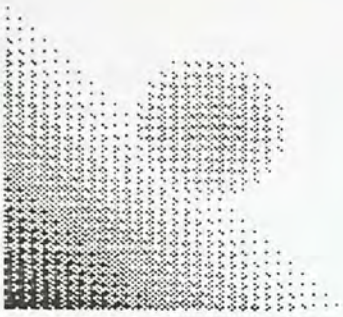
If the computed confidence η is above a predefined threshold T_η we accept the solution and stop. On the other hand if η is low, attempt is then made to get better results by modifying the parameters. If it turns out that all matches are nil, c_1 and c_2 will be lowered and the process is repeated.

If on the other hand the standard deviation in object reference points are greater than the predefined values, c_1 and c_2 are raised and the process is again repeated. Similar to the 2-D case the process is allowed to be repeated for at most 4 times. If by then the result is still no good, we claim that the object to be recognized is not present in the scene.

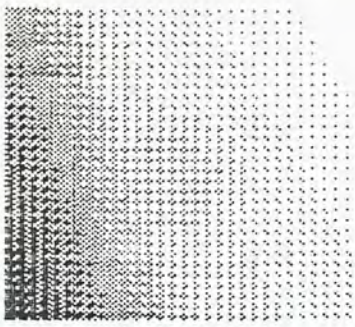
3.4 Experimental Results and Discussions

To verify the effectiveness of the algorithm, we first perform a preliminary test on the synthetic object shown in fig.3.4.1a, which is the same object in fig.3.2.7. Fig.3.4.1b shows a rotated version of the object which corresponds to the scene to be analysed. The resolution of the depth maps are 30 by 30 in this simple experiment. Three subtemplates are defined for the object, as shown in fig.3.4.1c. Since the rotation of the scene with respect to the model is known beforehand, the expected locations and orientations of the corresponding patches as well as the object reference points on the scene can be computed to check the correctness of the result. The table in fig.3.4.1d shows the locations as well as the orientations for the three subtemplates together with the three object reference points and their expected correspondence in the rotated scene. The match table is shown in fig.3.4.1e and the application of DP and the final result is shown in fig.3.4.1f. Entries marked with a '*' in the match table correspond to the final selected matches. Comparing them with the actual values shown in fig.3.4.1d we see that they are quite close and are therefore acceptable.

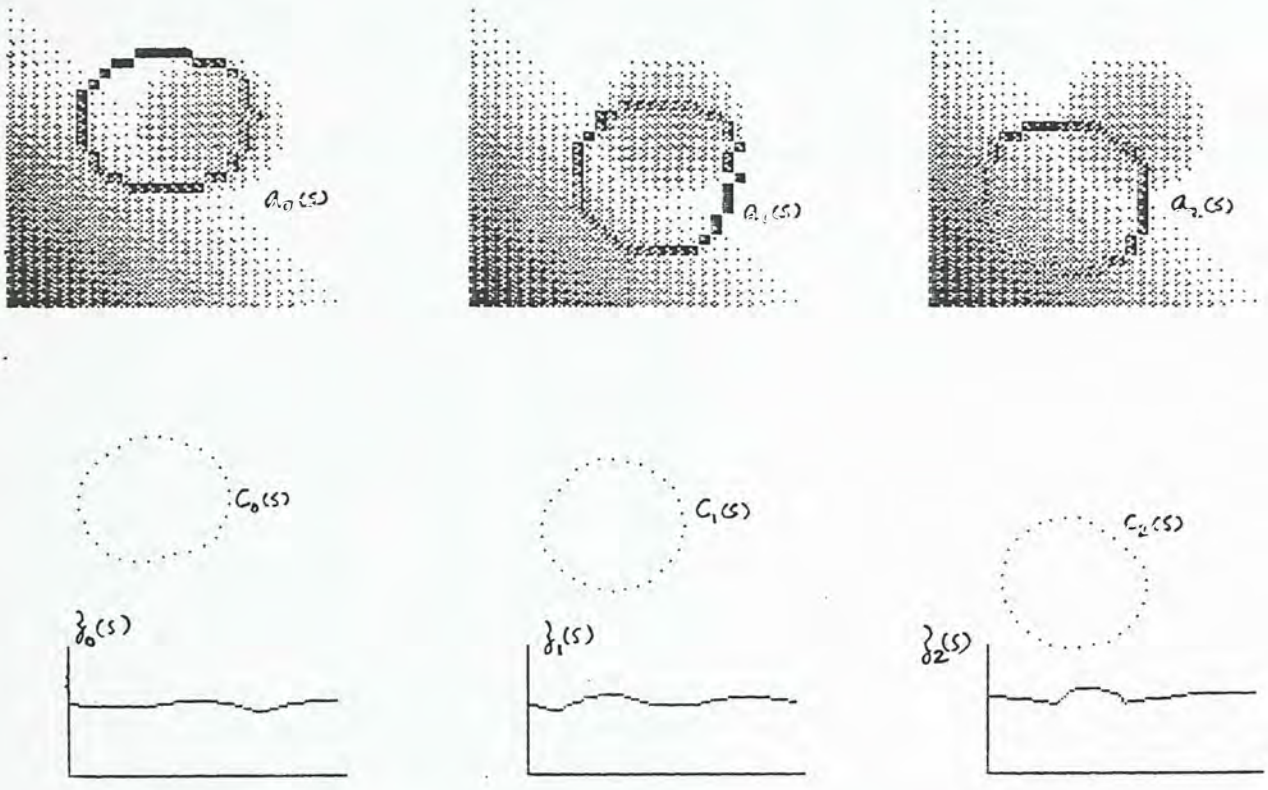
To study the behaviour of the algorithm when part of the object is being occluded, we perform the same experiment twice, with the matching region chosen in the second trial in such a way that one of the three subtemplates, namely subtemplate 0, should not be matched(ie. we avoid regions of correct match for that



(a) synthetic object



(b) rotated view



(c) Three subtemplates with their various projections

n	actual centre		
0	13.0000	11.0000	29.6569
1	16.0000	17.0000	28.7958
2	11.0000	19.0000	29.0000

n	orientation		
0	0.0769	0.2471	-0.9659
1	-0.2176	-0.0145	-0.9759
2	-0.2480	0.2215	-0.9431

	reference		
0	15.0000	15.0000	0.0000
1	15.0000	15.0000	15.0000
2	0.0000	0.0000	0.0000

n	actual centre on scene		
0	14.7240	11.0619	27.8710
1	14.2588	17.4266	30.1104
2	10.2564	15.5505	33.1931

n	actual orientation on scene		
0	-0.0461	0.5417	-0.5276
1	-0.7698	0.1755	-0.6137
2	-0.0483	0.3098	-0.4301

	actual reference on scene		
0	-4.1717	24.9777	9.1940
1	5.4142	19.9889	19.5970
2	-9.5858	4.9889	4.5970

(d) Table showing locations & orientation of the 3 subtemplates and their correspondence in the rotated scene

Fig 3.4.1 A simple experiment

		error	rotation	position	orientation	reference					
*	0 0	0.1298	-0.540	15,11,27.72	-0.630, 0.537,-0.561	-5.34, 24.75, 10.31	3.93, 19.85, 21.04	-10.55,	4.72,	5.67
	0 1	0.2615	-0.184	15,13,29.31	-0.669, 0.423,-0.612	-7.30, 20.56, 9.75	3.40, 17.94, 19.94	-15.62,	1.42,	13.57
	0 2	0.2795	-0.363	15,12,28.67	-0.637, 0.492,-0.593	-5.95, 23.00, 9.74	3.87, 19.10, 20.40	-13.05,	3.03,	8.98
	0 3	0.2887	2.951	15,14,29.74	-0.698, 0.354,-0.622	-7.50, 24.70, 12.58	1.72, 15.94, 20.53	6.74,	40.40,	13.24
	0 4	0.3214	-0.722	15,10,26.18	-0.619, 0.580,-0.530	-4.55, 26.56, 10.96	4.06, 20.55, 21.68	-7.58,	7.31,	2.59
	0 5	0.2539	1.560	15,15,30.00	-0.722, 0.294,-0.627	-10.82, 19.58, 13.58	2.02, 14.75, 19.65	-15.29,	30.14,	31.43
	1 0	0.1751	-0.849	14,17,30.20	-0.759, 0.196,-0.621	-2.24, 25.86, 8.22	7.36, 20.27, 18.31	-5.78,	6.32,	0.76
	1 1	0.2026	3.280	13,11,27.92	-0.660, 0.528,-0.534	-5.79, 32.71, 18.19	6.16, 25.53, 23.72	6.67,	48.79,	12.16
*	1 2	0.2029	-0.912	14,16,30.24	-0.745, 0.257,-0.615	-1.66, 27.50, 8.50	7.58, 21.07, 18.41	-6.20,	8.44,	0.37
	1 3	0.2130	-1.092	13,13,29.46	-0.692, 0.443,-0.570	-0.16, 33.76, 11.70	8.46, 24.40, 19.65	-2.05,	19.09,	-3.51
	1 4	0.2150	3.460	13,12,28.83	-0.676, 0.488,-0.551	-5.70, 32.55, 17.78	6.26, 25.39, 23.31	7.09,	46.38,	8.02
	1 5	0.2390	-0.942	13,15,30.13	-0.727, 0.329,-0.603	-1.16, 30.10, 9.61	7.90, 22.45, 18.80	-4.41,	12.54,	-1.83
	2 0	0.1567	-0.439	11,18,33.31	-0.859, 0.244,-0.450	-4.54, 22.86, 4.17	5.47, 20.09, 14.99	-11.27,	2.77,	5.26
*	2 1	0.1902	-0.767	10,15,33.66	-0.852, 0.320,-0.414	-3.37, 30.25, 5.77	6.22, 24.60, 15.82	-7.95,	10.64,	-0.90
	2 2	0.2374	-0.610	12,16,30.08	-0.744, 0.297,-0.599	1.46, 26.64, 2.10	8.82, 22.31, 14.43	-4.10,	6.52,	-1.65
	2 3	0.2569	-0.708	10,14,31.93	-0.798, 0.377,-0.470	-0.96, 31.11, 4.85	7.51, 24.84, 15.52	-6.50,	11.84,	-2.07
	2 4	0.2599	-1.052	11,13,28.94	-0.703, 0.460,-0.543	3.76, 37.56, 7.18	10.15, 28.69, 17.45	3.03,	21.29,	-6.41
	2 5	0.2755	-1.023	11,12,28.25	-0.687, 0.500,-0.527	4.38, 38.37, 7.69	10.47, 29.00, 17.70	2.99,	22.49,	-6.31

Fig 3.4.1e The match table. Entries marked with '*' are the selected results.

```
Total no. of subtemplates :  3
no. of matched subtemplates :  3
compatability (0,h,1,1)
 l  h   -1       0       1       2       3       4       5
-1:   0.7034  0.5190  0.5190  0.5190  0.5190  0.5190  0.5190
 0:   0.5190  0.9317  0.5390  0.6747  0.4254  0.8766  0.3876
 1:   0.5190  0.3647  0.3945  0.3661  0.5185  0.3888  0.4815
 2:   0.5190  0.7392  0.4880  0.5899  0.3997  0.9415  0.3666
 3:   0.5190  0.4589  0.3850  0.4163  0.3985  0.5473  0.3955
 4:   0.5190  0.3907  0.3658  0.3929  0.4918  0.4229  0.4202
 5:   0.5190  0.6025  0.4290  0.5088  0.3891  0.9030  0.3675
compatability (1,h,2,1)
 l  h   -1       0       1       2       3       4       5
-1:   0.7351  0.4618  0.4618  0.4618  0.4618  0.4618  0.4618
 0:   0.4618  0.7444  0.4199  0.7456  0.5900  0.4667  0.6992
 1:   0.4618  0.8966  0.4992  1.0447  0.8497  0.5789  0.8652
 2:   0.4618  0.6778  0.7119  0.6541  0.5617  0.5519  0.5779
 3:   0.4618  0.9130  0.6425  0.8503  0.6727  0.6054  0.8189
 4:   0.4618  0.7342  0.4638  0.6284  0.5961  0.4407  0.5823
 5:   0.4618  0.6198  0.4343  0.8186  0.6000  0.4337  0.6139
compatability (2,h,0,1)
 l  h   -1       0       1       2       3       4       5
-1:   0.6785  0.5662  0.5662  0.5662  0.5662  0.5662  0.5662
 0:   0.5662  0.7600  0.7585  0.5918  0.6329  0.4254  0.4074
 1:   0.5662  0.7324  0.5662  0.4519  0.4781  0.3848  0.3919
 2:   0.5662  0.9050  0.6992  0.5130  0.5405  0.4016  0.3929
 3:   0.5662  0.5029  0.4843  0.3873  0.4337  0.4052  0.4221
 4:   0.5662  0.5989  0.6791  0.7511  0.7901  0.4753  0.4419
 5:   0.5662  0.4617  0.4573  0.3643  0.4226  0.4028  0.4290
template :     0    1    2
match no :     0    2    1
ref. point              ø                       1.                      2
   0 0 0 *  -5.34  24.75  10.31 *   3.93  19.85  21.04 *  -10.55   4.72   5.67
   1 1 2 *  -1.66  27.50   8.50 *   7.58  21.07  18.41 *   -6.20   8.44   0.37
   2 2 1 *  -3.37  30.25   5.77 *   6.22  24.60  15.82 *   -7.95  10.64  -0.90
mean ref. *  -3.46  27.50   8.19 *   5.91  21.84  18.42 *   -8.23   7.93   1.71
standard deviation : 6.2398
```

Fig 3.4.1f Result of the DP scheme

Fig 3.4.1 Cont'd

	error	rotation	position	orientation	reference		
0 0	0.2987	2.951	15,14,29.74	-0.698, 0.354,-0.622	-7.50, 24.70, 12.58	1.72, 15.94, 20.53	1.99, 35.16, 13.09
0 1	0.3529	1.560	15,15,30.00	-0.722, 0.294,-0.627	-10.82, 19.58, 13.58	2.02, 14.75, 19.65	-13.80, 26.62, 25.48
0 2	0.4141	3.103	15,15,30.10	-0.738, 0.234,-0.633	-8.48, 21.35, 11.47	1.14, 14.23, 20.51	1.29, 31.23, 8.84
1 0	0.1751	-0.849	14,17,30.20	-0.759, 0.196,-0.621	-2.24, 25.86, 8.22	7.36, 20.27, 18.31	-4.60, 12.83, 3.25
1 1	0.2029	-0.812	14,15,30.24	-0.745, 0.257,-0.615	-1.66, 27.50, 8.50	7.58, 21.07, 18.41	-4.69, 14.79, 3.08
1 2	0.2390	-0.942	13,15,30.13	-0.727, 0.329,-0.603	-1.16, 30.10, 9.61	7.90, 22.45, 18.80	-3.33, 18.39, 1.99
1 3	0.2580	-1.139	13,14,29.88	-0.703, 0.376,-0.603	-0.70, 32.02, 10.64	8.26, 23.54, 19.17	-1.13, 21.77, 0.90
1 4	0.2791	-0.702	15,18,29.86	-0.775, 0.132,-0.619	-2.63, 23.25, 7.63	7.08, 18.92, 18.21	-5.94, 9.73, 5.13
1 5	0.2811	-0.673	14,18,30.00	-0.779, 0.143,-0.611	-2.74, 23.43, 7.79	7.00, 19.00, 18.29	-6.37, 9.96, 5.47
2 0	0.1567	-0.439	11,18,33.31	-0.859, 0.244,-0.450	-4.54, 22.86, 4.17	5.47, 20.09, 14.99	-9.03, 9.46, 4.90
2 1	0.1902	-0.767	10,15,33.66	-0.852, 0.320,-0.414	-3.37, 30.25, 5.77	6.22, 24.60, 15.82	-6.42, 17.18, 1.32
2 2	0.2374	-0.610	12,16,30.08	-0.744, 0.297,-0.599	1.46, 26.64, 2.10	8.82, 22.31, 14.43	-2.25, 13.23, -0.40
2 3	0.2589	-0.708	10,14,31.93	-0.798, 0.377,-0.470	-0.96, 31.11, 4.85	7.51, 24.84, 15.52	-4.66, 18.26, 0.24
2 4	0.3007	-0.441	12,17,30.03	-0.757, 0.248,-0.604	0.35, 22.61, 1.68	8.15, 19.90, 14.20	-4.45, 9.31, 1.79
2 5	0.3165	-0.339	13,18,30.00	-0.777, 0.161,-0.609	-0.67, 18.29, 1.87	7.58, 17.51, 14.37	-5.66, 5.29, 4.35

(a) The match table

```
Total no. of subtemplates :  3
no. of matched subtemplates :  3
compatability (0,h,1,1)
 l  h   -1       0       1       2
-1:   0.7024  0.5190  0.5190  0.5190
 0:   0.5190  0.4585  0.4024  0.4153
 1:   0.5190  0.4302  0.3853  0.4075
 2:   0.5190  0.4198  0.3854  0.4300
 3:   0.5190  0.4143  0.3905  0.4512
 4:   0.5190  0.4029  0.4378  0.4307
 5:   0.5190  0.4061  0.4405  0.4329
compatability (1,h,2,1)
 l  h   -1       0       1       2       3       4       5
-1:   0.7361  0.4618  0.4618  0.4618  0.4618  0.4618  0.4618
 0:   0.4618  0.7805  0.7694  0.7175  0.7576  1.0550  0.9091
 1:   0.4618  0.2971  1.0345  0.8860  0.8312  0.6638  0.7291
 2:   0.4618  0.6517  0.6250  0.5809  0.5579  0.6565  0.6172
 3:   0.4618  0.3105  0.3441  0.3174  0.7234  0.6811  0.7449
 4:   0.4618  0.6425  0.5860  0.5461  0.5481  0.7117  0.6681
 5:   0.4618  0.5435  0.5428  0.5420  0.5821  0.6254  0.5951
compatability (2,h,0,1)
 l  h   -1       0       1       2       3       4       5
-1:   0.6765  0.5662  0.5662  0.5662  0.5662  0.5662  0.5662
 0:   0.5662  0.5263  0.5096  0.4057  0.4565  0.4224  0.4413
 1:   0.5662  0.4810  0.4733  0.3763  0.4370  0.3933  0.4140
 2:   0.5662  0.4717  0.5011  0.4037  0.4790  0.3980  0.4024
Template :     0     1     2
Match no :    -1     4     0
Ref. point                  1                       2
    1 1 4 *  -2.63  23.25   7.63 *   7.08  18.92  18.21 *  -5.94   9.73   5.13
    2 2 0 *  -4.54  22.86   4.17 *   5.47  20.09  14.99 *  -9.03   9.46   4.90
mean ref. *  -3.59  23.05   5.90 *   6.27  19.50  16.60 *  -7.48   9.60   5.01
standard deviation : 3.1535
```

(b) Result of the DP scheme

Fig 3.4.2 Same experiment as in Fig.3.4.1 to illustrate occlusion

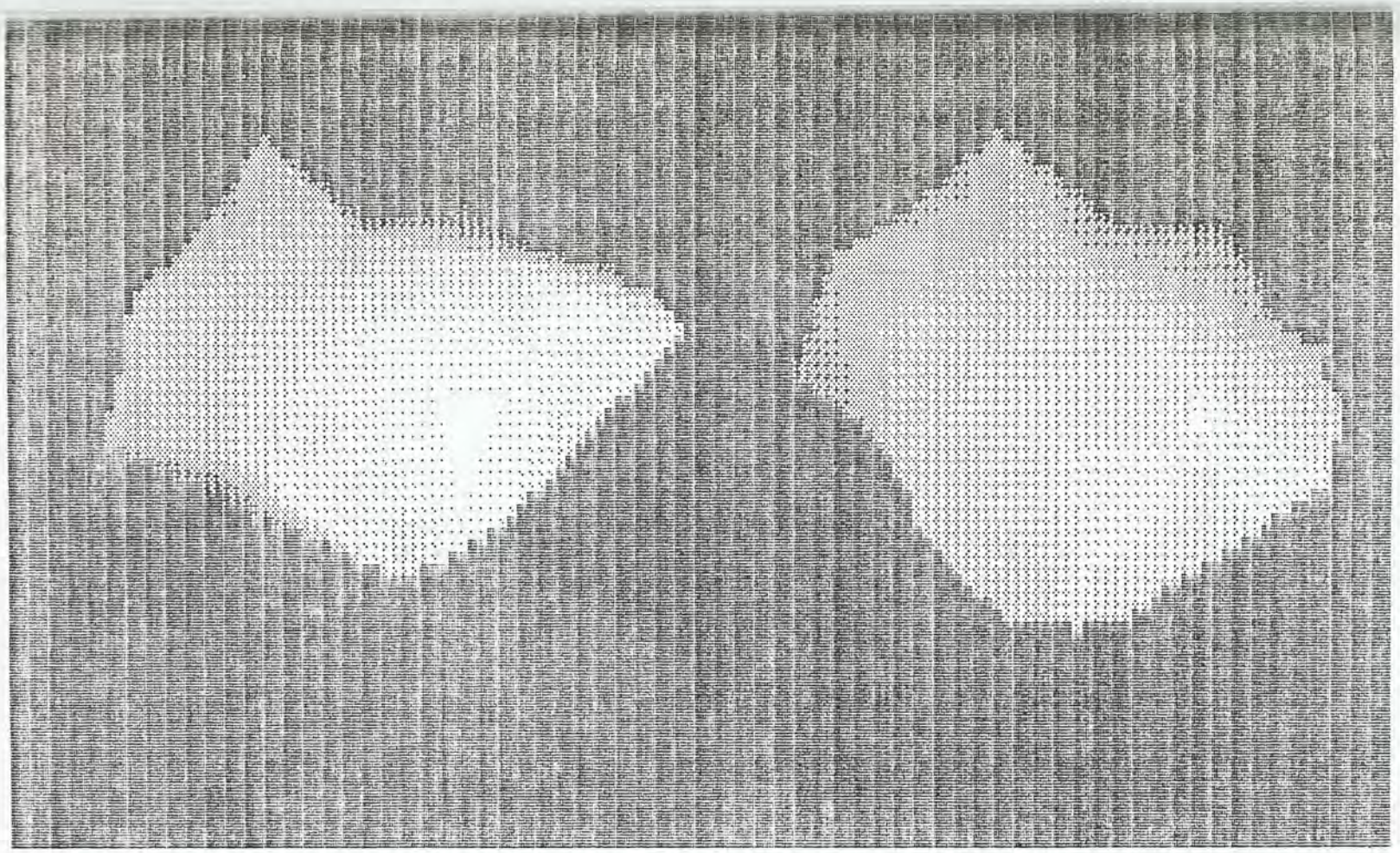
subtemplate), just as if it were being occluded. Fig.3.4.2a shows the resulting match table. There are still two false matches for subtemplate 0, but are rejected by the DP scheme as shown in fig.3.4.2b. From the figure it can be seen that subtemplate 0 is labeled as -1, which means a nul label. The remaining two selected matches for subtemplate 1 and 2 are the same as those in the previous experiment.

After that another experiment is performed on a more complex object, an arbitrarily-shaped stone as shown in fig.3.4.3. Since we do not have a range-finder, the object is approximated by and manually digitized into a polyhedra consisting some 106 triangular faces. From that synthetic depth maps as well as intensity maps are generated for different viewing directions. The size of the maps are 128 by 128. Note that from this figure on, bright regions on the picture correspond to regions close to the viewer and vice versa. A library is then constructed for the object based on the various depth maps. For simplicity but without loss of generality, we used only one depth map in constructing the library. That means the subtemplates are confined to one side of the stone. In reality one needs some four or five views in order to cover the entire object. However for illustration purposes, using one will suffice. In selecting a subtemplate one should try the best to obtain it from a viewing direction so that it is completely visible. A subtemplate that cannot be completely seen no matter at which direction it is viewed should be avoided.

The depth map of the model at the specific view from which the subtemplates are extracted is shown in fig.3.4.5a. The

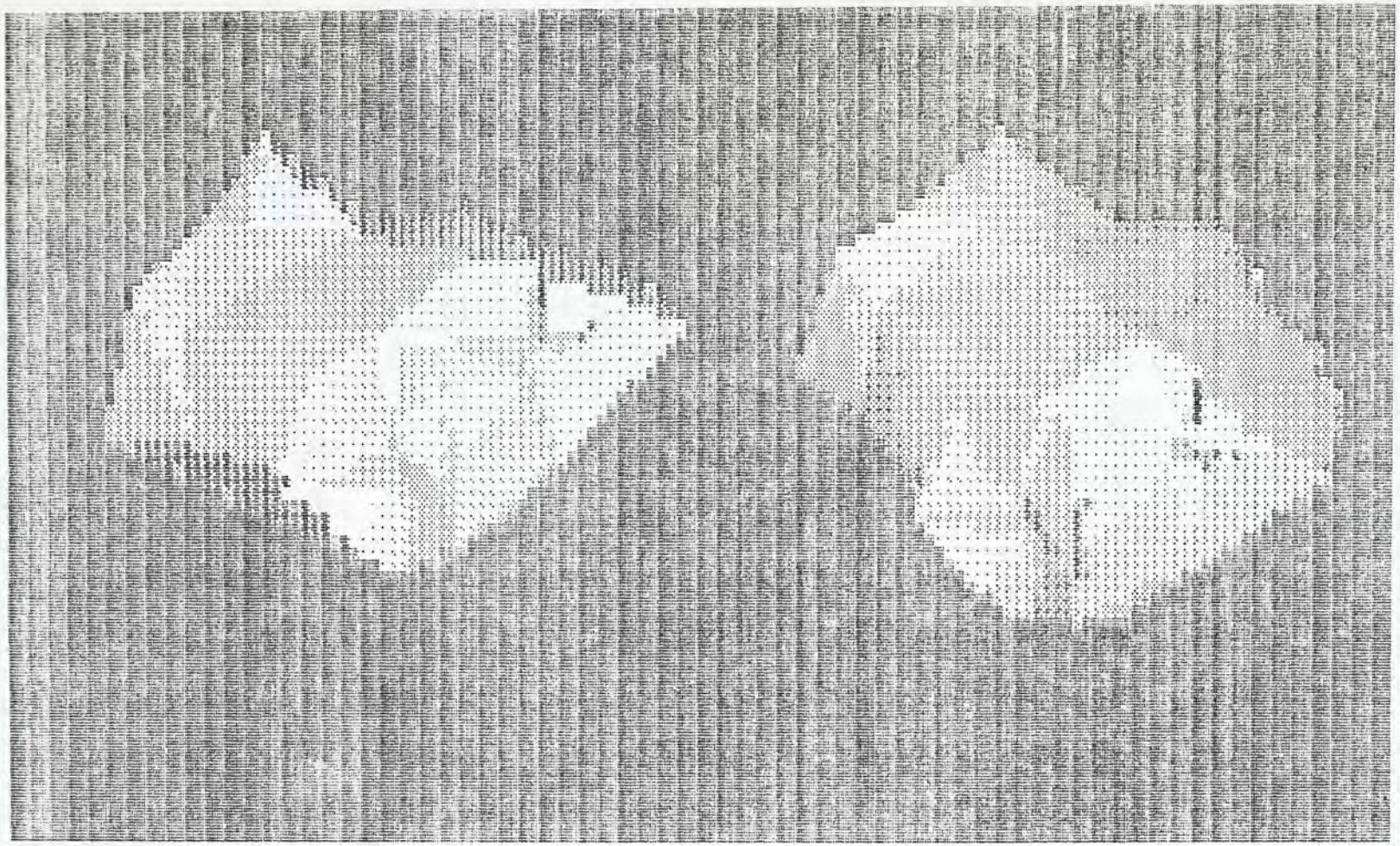
extracted subtemplates are shown with the obtained space curves in heavy lines. The table in fig.3.4.3b records the locations and orientations for each subtemplate as well as the defined object reference points. The scene to be analyzed is derived from the rotated stone shown in fig.3.4.3b. The actual location and orientation of matching for each subtemplate and the actual object reference points on the scene are shown in fig.3.4.5c. Fig.3.4.5d shows the match table, with the calculated compatibility used in the DP scheme shown in fig.3.4.5e. Fig.3.4.5f shows the final result. Compared with the expected values in 3.4.5c, we see that they are quite close.

From the results it can be seen that the error in matching position and orientation are quite small compared to that of the object reference points. The reason is that since the object reference points are in general quite far from the subtemplates, a little deviation in the matching position and orientation will cause the error to be amplified by a great factor, resulting in a relatively large deviation from the expected locations of the object reference points. Therefore it may be necessary to modify the way the reference point locations are inferred, say by directly calculating them in a global manner from the resulting matching positions alone. The idea is we estimate the location of the object reference points based on all matched subtemplates, instead of calculating them based on each individual subtemplate and then average.



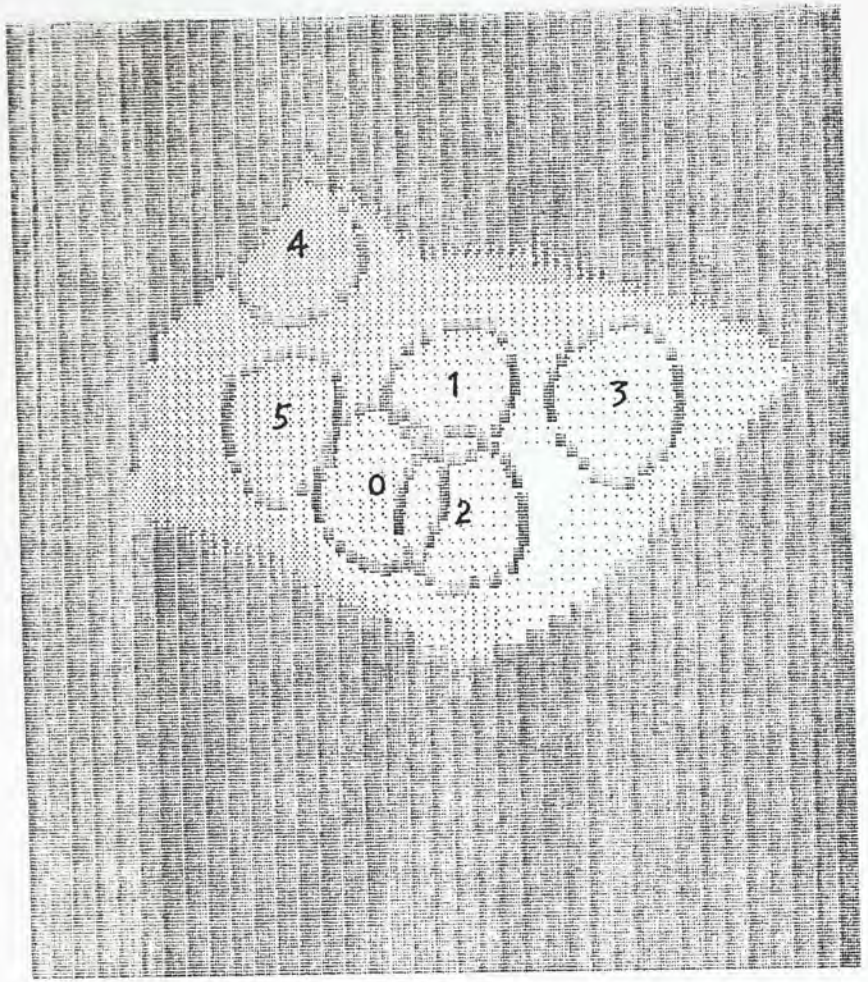
(a) (b)

Fig 3.4.3 Depth maps of a synthetic stone at two different viewing directions



(a) (b)

Fig 3.4.4 Intensity image of a synthetic stone at two different viewing directions



(a)

n	actual centre		
0	32.0000	34.0000	29.0000
1	38.0000	26.0000	32.0000
2	39.0000	36.0000	32.0000
3	53.0000	28.0000	32.0000
4	25.0000	17.0000	16.0000
5	23.0000	29.0000	25.0000

n	orientation		
0	0.2226	0.0516	-0.9736
1	0.2566	0.4104	-0.8751
2	0.3389	-0.1726	-0.9249
3	-0.1772	0.1609	-0.9709
4	0.4609	0.4614	-0.7581
5	0.5423	0.3190	-0.7773

n	reference		
0	40.0000	30.0000	20.0000
1	50.0000	30.0000	20.0000
2	40.0000	40.0000	20.0000

(b)

n	expected centre on scene		
0	27.5847	37.0710	27.6557
1	34.1036	31.5945	33.6981
2	33.0658	41.3193	31.3892
3	47.9172	37.2564	36.1736
4	27.5527	16.6269	18.2197
5	21.0535	29.2798	23.3353

n	expected orientation on scene		
0	0.4451	-0.0865	-0.8911
1	0.3814	0.2755	-0.8812
2	0.5876	-0.2620	-0.7855
3	0.0440	-0.0848	-0.9954
4	0.5354	0.4024	-0.7426
5	0.6457	0.2841	-0.7088

n	expected reference on scene		
0	38.2180	33.5040	21.7220
1	47.6918	36.0135	23.7072
2	36.2326	42.9778	19.2106

(c)

Fig 3.4.5 A more complicated experiment.
(a) Model objects and subtemplates;
(b) Centre, orientation of subtemplates and reference points;
(c) Expected centre, orientation and reference points on scene.

	error	rotation	position	orientation	reference ϕ	1	2
0 0 0	0.1413	3.157	28,19,19.00	0.385, 0.439,-0.807	30.39, 43.19, 20.40	22.34, 45.29, 14.85	31.46, 34.51, 15.37
0 0 1	0.1713	-0.392	28,37,28.00	0.449,-0.099,-0.888	42.94, 31.21, 23.20	52.09, 34.85, 24.92	39.93, 40.23, 20.09
0 0 2	0.1846	3.161	27,18,18.00	0.414, 0.461,-0.785	30.60, 43.45, 20.64	22.69, 45.74, 14.98	31.76, 34.90, 15.58
0 0 3	0.2693	3.643	30,22,22.00	0.351, 0.557,-0.753	32.14, 45.64, 23.95	24.48, 51.73, 21.92	28.89, 39.22, 17.00
0 0 4	0.2710	2.083	29,21,21.00	0.366, 0.504,-0.783	29.81, 37.45, 16.99	27.03, 31.81, 9.21	38.58, 32.65, 17.34
0 0 5	0.2919	-1.144	39,18,14.00	-0.478, 0.825,-0.303	33.47, 46.13, 25.60	38.45, 53.44, 30.27	26.94, 45.75, 33.16
1 1 0	0.1118	-0.295	34,32,34.00	0.383, 0.256,-0.887	42.10, 27.62, 19.98	51.63, 29.98, 21.68	40.29, 37.08, 17.30
1 1 1	0.1935	-0.108	32,32,33.00	0.439, 0.209,-0.874	42.54, 27.20, 20.09	52.31, 27.74, 22.18	42.52, 36.91, 17.70
1 1 2	0.2180	-0.261	34,33,34.00	0.418, 0.195,-0.887	42.49, 26.84, 20.03	52.07, 28.84, 22.10	41.21, 36.25, 16.88
1 1 3	0.2181	-0.163	33,31,33.00	0.396, 0.313,-0.863	42.09, 28.53, 20.13	51.84, 29.65, 22.05	41.26, 38.37, 18.57
1 1 4	0.2393	-0.300	35,32,34.00	0.357, 0.288,-0.888	41.78, 28.04, 19.93	51.52, 30.48, 21.68	39.77, 37.55, 17.60
1 1 5	0.2579	2.796	44,27,33.00	-0.036, 0.711,-0.702	38.39, 36.26, 24.34	29.00, 35.01, 20.92	41.73, 33.34, 15.38
2 2 0	0.1290	-1.117	21,25,21.00	0.484, 0.392,-0.782	47.25, 36.44, 21.38	51.31, 45.21, 28.96	38.35, 40.89, 20.31
2 2 1	0.1465	0.016	31,42,30.00	0.438,-0.228,-0.869	41.24, 29.33, 20.53	51.18, 29.20, 21.68	41.43, 39.31, 20.00
2 2 2	0.1503	3.573	21,30,23.00	0.690, 0.243,-0.682	50.69, 40.57, 27.17	46.45, 45.79, 19.77	47.24, 32.08, 23.18
2 2 3	0.1536	3.108	34,22,24.00	0.100, 0.703,-0.705	43.25, 47.41, 26.28	34.29, 49.74, 22.51	43.31, 39.20, 20.58
2 2 4	0.1585	3.551	22,30,24.00	0.638, 0.287,-0.696	50.35, 41.14, 26.93	45.68, 46.38, 19.81	47.08, 32.63, 22.82
2 2 5	0.1629	-1.002	26,25,24.00	0.362, 0.592,-0.720	45.44, 39.36, 20.67	50.10, 47.20, 24.76	36.61, 43.77, 22.26
3 3 0	0.1298	1.631	23,26,23.00	0.466, 0.499,-0.731	58.65, 44.16, 27.10	58.76, 34.91, 23.28	68.11, 43.01, 30.14
3 3 1	0.1588	-0.231	47,40,36.00	-0.083,-0.214,-0.973	41.56, 22.52, 19.51	51.19, 25.15, 19.93	39.23, 31.61, 16.05
3 3 2	0.1694	1.601	23,28,24.00	0.547, 0.412,-0.729	59.57, 43.45, 26.08	59.76, 33.87, 23.23	68.66, 42.43, 30.12
3 3 3	0.1746	-0.225	47,39,36.00	-0.056,-0.168,-0.984	41.90, 23.18, 18.94	51.53, 25.73, 19.76	39.71, 32.44, 15.86
3 3 4	0.1863	1.801	27,27,26.00	0.397, 0.568,-0.721	59.38, 44.25, 28.52	58.03, 35.71, 23.50	69.03, 41.97, 29.80
3 3 5	0.1871	1.667	21,25,21.00	0.484, 0.392,-0.782	59.55, 43.19, 25.42	59.04, 33.62, 22.57	68.96, 41.78, 28.48
4 4 0	0.1287	-0.292	28,18,19.00	0.429, 0.413,-0.803	36.39, 33.58, 18.30	46.11, 35.87, 18.93	34.27, 43.15, 16.34
4 4 1	0.1622	2.833	23,23,20.00	0.331, 0.535,-0.777	21.00, 10.62, -2.80	14.47, 12.68,-10.09	26.91, 5.99, -9.40
4 4 2	0.1751	-0.288	27,17,18.00	0.464, 0.458,-0.758	36.26, 33.40, 19.77	45.93, 35.64, 20.95	34.21, 43.08, 18.28
4 4 3	0.1857	1.040	48,40,35.00	-0.052,-0.280,-0.959	41.20, 8.54, 7.28	44.42, -0.64, 4.97	47.87, 12.47, 0.95
4 4 4	0.1870	1.071	48,39,35.00	-0.008,-0.210,-0.978	41.43, 8.73, 7.53	44.55, -0.35, 4.74	48.52, 12.91, 1.85
4 4 5	0.1871	2.905	21,23,19.00	0.355, 0.429,-0.831	20.01, 9.25, -2.03	13.49, 11.17, -9.37	25.34, 3.54, -8.26
5 5 0	0.0876	-0.254	22,29,24.00	0.623, 0.332,-0.708	40.25, 32.63, 22.93	49.92, 34.51, 24.66	38.60, 42.40, 21.55
5 5 1	0.0913	-0.425	21,29,23.00	0.662, 0.281,-0.695	39.99, 34.00, 23.88	49.24, 37.08, 26.12	37.66, 43.23, 20.80
5 5 2	0.0938	-0.802	35,23,26.00	0.116, 0.723,-0.681	33.72, 42.76, 21.72	40.54, 50.02, 22.63	26.41, 49.47, 22.92
5 5 3	0.1011	2.695	24,25,23.00	0.402, 0.547,-0.734	18.56, 35.42, 9.06	13.88, 37.13, 0.39	25.35, 29.85, 4.29
5 5 4	0.1060	-0.402	22,31,25.00	0.677, 0.242,-0.695	40.16, 33.40, 23.93	49.49, 36.22, 26.19	38.17, 42.62, 20.62
5 5 5	0.1126	-0.703	38,24,27.00	0.113, 0.752,-0.649	34.10, 42.49, 21.09	41.24, 49.44, 22.82	27.13, 49.41, 23.78

Fig 3.4.5 (d) The match table.

compatability (0,h,1,1)

l h	-1	0	1	2	3	4	5
-1:	0.6934	0.4818	0.4818	0.4818	0.4818	0.4818	0.4818
0:	0.4818	0.3687	0.3857	0.3701	0.4185	0.4128	0.2740
1:	0.4818	0.3735	0.3752	0.3731	0.4295	0.4350	0.2636
2:	0.4818	0.3780	0.7489	0.3795	0.4225	0.4298	0.2637
3:	0.4818	0.3615	0.3234	0.3605	0.4209	0.4148	0.2793
4:	0.4818	0.3636	1.0070	0.3654	0.4072	0.4107	0.2902
5:	0.4818	0.4313	0.3545	0.4282	0.4419	0.4362	0.3996

compatability (1,h,2,1)

l h	-1	0	1	2	3	4	5
-1:	0.5959	0.4763	0.4763	0.4763	0.4763	0.4763	0.4763
0:	0.4763	0.4057	0.4019	0.3976	0.4129	0.4095	0.4293
1:	0.4763	1.1185	1.0977	1.1194	1.4804	1.1389	0.3658
2:	0.4763	0.3939	0.4005	0.3854	0.4084	0.3957	0.3736
3:	0.4763	0.4555	0.4788	0.4564	0.4790	0.4461	0.4203
4:	0.4763	0.3977	0.4163	0.3906	0.4150	0.3956	0.3850
5:	0.4763	0.4371	0.4700	0.4353	0.4545	0.4232	0.3973

compatability (2,h,3,1)

l h	-1	0	1	2	3	4	5
-1:	0.6409	0.5291	0.5291	0.5291	0.5291	0.5291	0.5291
0:	0.5291	0.3107	0.4466	0.2949	0.3946	0.3503	0.3104
1:	0.5291	0.3732	0.7023	0.3511	0.3418	0.3538	0.3509
2:	0.5291	0.3066	0.5604	0.3374	0.4211	0.3260	0.3277
3:	0.5291	0.3910	0.7039	0.3524	0.3556	0.3647	0.3643
4:	0.5291	0.3378	0.3566	0.3965	0.3500	0.3760	0.3959
5:	0.5291	0.3908	0.4388	0.3429	0.4510	0.3416	0.3289

compatability (3,h,4,1)

l h	-1	0	1	2	3	4	5
-1:	0.6504	0.5494	0.5494	0.5494	0.5494	0.5494	0.5494
0:	0.5494	0.2671	0.6303	0.2593	0.5189	0.2747	0.2627
1:	0.5494	0.2312	0.5535	0.2463	0.5062	0.2253	0.2400
2:	0.5494	0.2628	0.5635	0.2589	0.6460	0.2731	0.2650
3:	0.5494	0.3208	0.2614	0.3618	0.3652	0.3216	0.6424
4:	0.5494	0.2869	0.2627	0.2748	0.2605	0.2864	0.5696
5:	0.5494	0.2202	0.4212	0.2290	0.5485	0.2243	0.2299

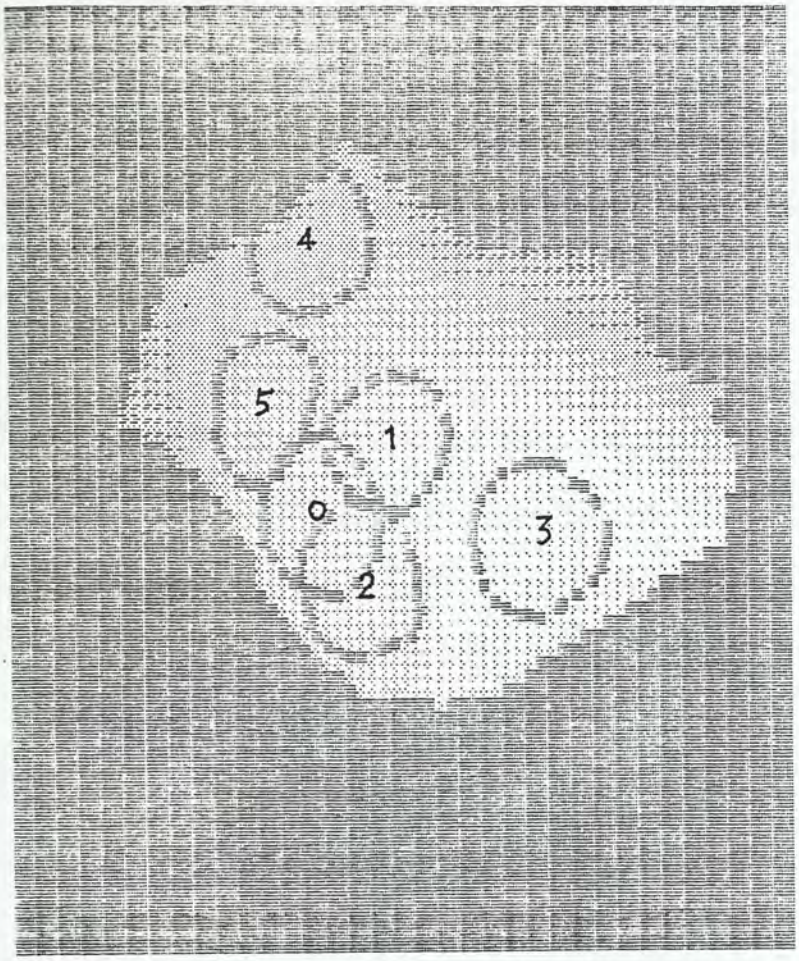
compatability (4,h,5,1)

l h	-1	0	1	2	3	4	5
-1:	0.6670	0.5233	0.5233	0.5233	0.5233	0.5233	0.5233
0:	0.5233	0.6770	0.3018	0.7991	0.2423	0.2520	0.3500
1:	0.5233	0.6682	0.3175	0.7838	0.2393	0.2462	0.3333
2:	0.5233	0.4430	0.4472	0.4095	0.2415	0.2524	0.5421
3:	0.5233	0.4118	0.3674	0.4097	0.2367	0.2456	0.3943
4:	0.5233	0.7930	0.3275	0.7139	0.2446	0.2535	0.3477
5:	0.5233	0.4230	0.5944	0.6187	0.2492	0.2520	0.4118

compatability (5,h,0,1)

l h	-1	0	1	2	3	4	5
-1:	0.6684	0.4896	0.4896	0.4896	0.4896	0.4896	0.4896
0:	0.4896	0.4936	0.5017	0.7048	0.4318	0.4521	0.5855
1:	0.4896	1.1431	0.9016	0.3705	0.4068	0.8913	0.3766
2:	0.4896	0.4572	0.4752	0.5300	0.4569	0.4989	0.5161
3:	0.4896	0.9333	0.6771	0.4681	0.3695	0.5462	0.5158
4:	0.4896	0.7665	0.5574	0.4649	0.3946	0.5027	0.5406
5:	0.4896	0.2089	0.2307	0.5502	0.2943	0.2919	0.5402

Fig 3.4.5 (e) The compatability table



sub-template :		0		1		2		3		4		5							
Match no :		1		3		1		3		2		0							
		position			orientation					reference		ϕ		1			2		
0	1	28,37,28.00			0.449,-0.099,-0.888			*	42.94	31.21	23.20	*	52.09	34.85	24.92	*	39.93	40.23	20.09
1	3	33.31,33.00			0.396, 0.313,-0.863			*	42.09	28.58	20.13	*	51.84	29.65	22.05	*	41.26	38.37	18.57
2	1	31,42,30.00			0.438,-0.228,-0.869			*	41.24	29.33	20.53	*	51.18	29.20	21.68	*	41.43	39.31	20.00
3	3	47,39,36.00			-0.056,-0.168,-0.984			*	41.90	28.18	18.94	*	51.53	25.73	19.75	*	39.71	32.44	15.86
4	2	27,17,18.00			0.464, 0.458,-0.758			*	36.26	33.40	19.77	*	45.93	35.64	20.95	*	34.21	43.08	18.28
5	0	22,29,24.00			0.623, 0.332,-0.708			*	40.25	32.68	22.93	*	49.92	34.51	24.66	*	39.60	42.40	21.55
mean ref.		40.78	29.71	20.92	50.41	31.60	22.04	33.19	39.30	19.06									
standard deviation : 7.8230																			

Fig 3.4.5 (f) The final result. The selected matches are shown on the scene.

Fig 3.4.6 shows another example. The stone viewed at another direction is used as the scene. Fig.3.4.6b shows the match table. Note that subtemplate number 4 is missing, as shown in the final result in fig.3.4.6c and fig.3.4.6d. Compare the results with the actual values pre-computed, we see that the stone is correctly located.

To study the effect of partial occlusion, another two experiments were done on the scene shown in fig.3.4.3b and fig.3.4.6a by introducing a tetrahedra to occlude part of the stone, as shown in fig.3.4.7a and fig.3.4.8a respectively. The results are shown in fig.3.4.7b-c and fig.3.4.8b-c. They are all reasonable and correct.

Throughout the experiment the depth maps are manipulated and stored in integer values. The round-off error results in an inherent random noise of about 3%. To study the robustness of the scheme, we explicitly introduce some random noise of 6% on the scene in fig.3.4.6a. The resulting match table is shown in fig.3.4.9b. Compare that with the match table in fig.3.4.6b, we see that 4 out of 6 subtemplates cannot be matched, and the final result consists of two matched subtemplates, as shown in fig.3.4.9c. Although there are only two successfully matched subtemplates, the result is still correct.

Note that in reality the data obtained by a range-finder can be very accurate, an error of less than 3% is not uncommon. Anyway, to improve the situation with presence of even more severe random noise, one may increase the size of the match table

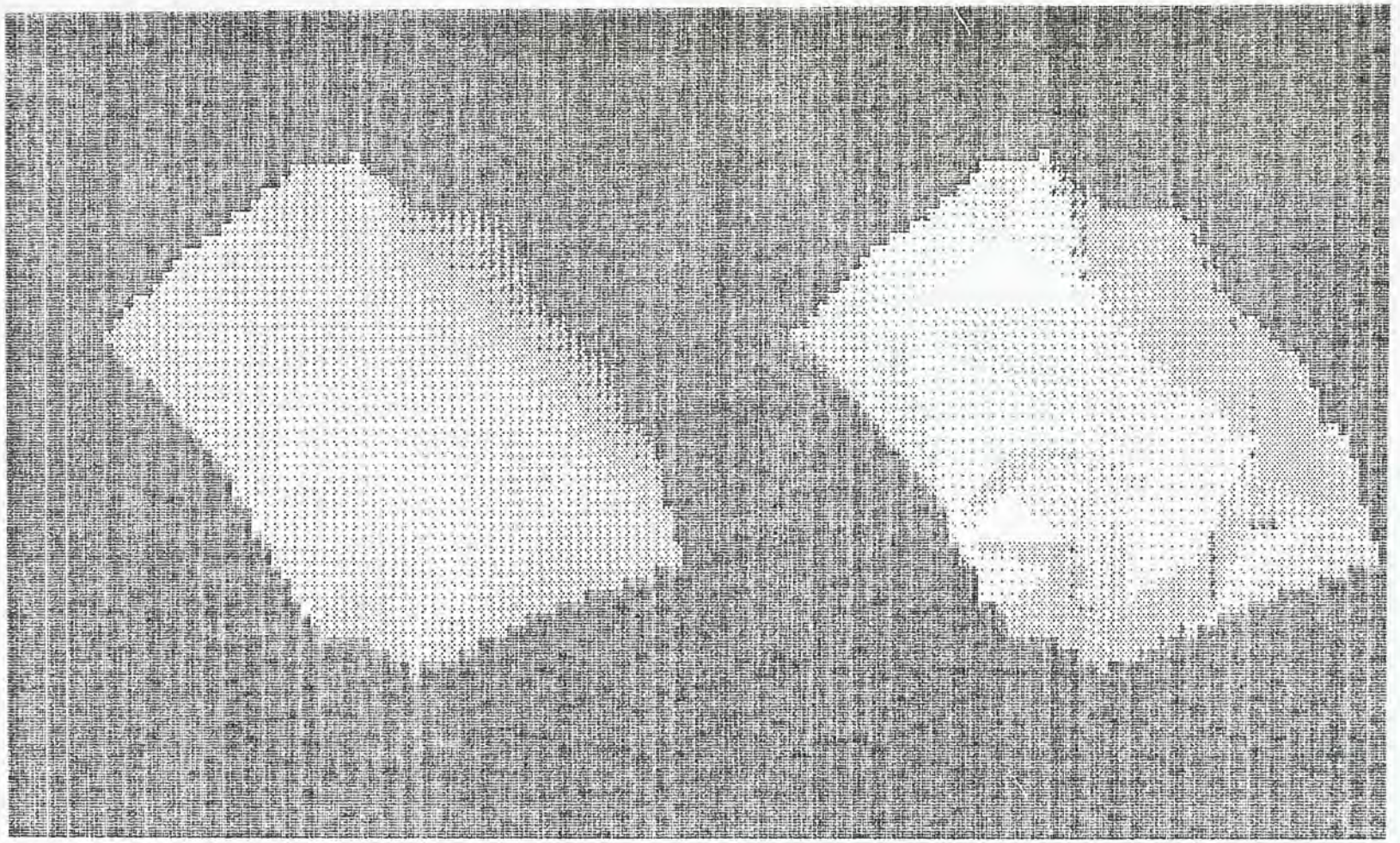


Fig 3.4.6 (a) Another view of the synthetic stone used for matching as a scene.
Left : depth image; right : intensity image.

	error	rotation	position	orientation	reference	0	1	2
0 0 0	0.1230	-0.461	32,40,26.00	0.069,-0.332,-0.941	39.90, 29.29, 20.25	48.84, 32.70, 17.35	35.55, 37.48, 16.50	
0 0 1	0.2040	-0.257	54,17,-4.00	-0.251, 0.168,-0.953	35.90, 33.04, 16.96	44.53, 36.00, 12.88	33.47, 42.58, 18.74	
0 0 2	0.2205	0.526	37,46,24.00	0.233,-0.466,-0.854	36.70, 23.33, 24.00	45.08, 18.41, 26.37	42.15, 31.03, 20.68	
0 0 3	0.2650	2.386	30,23,23.00	-0.111, 0.343,-0.933	23.56, 36.73, 19.93	16.09, 31.11, 16.37	30.21, 30.25, 16.20	
0 0 4	0.2684	3.550	33,21,22.00	-0.079, 0.194,-0.978	27.69, 42.30, 20.43	18.65, 46.52, 19.72	23.68, 33.36, 18.45	
0 0 5	0.2723	2.387	32,24,23.00	-0.114, 0.354,-0.928	23.53, 36.85, 19.99	16.04, 31.28, 16.39	30.16, 30.41, 16.17	
1 1 0	0.1882	-0.401	40,36,30.00	-0.040, 0.050,-0.998	36.78, 24.81, 19.31	45.77, 28.19, 16.55	32.55, 33.16, 15.78	
1 1 1	0.1924	-0.215	41,36,30.00	-0.087, 0.072,-0.994	35.85, 25.25, 19.40	45.14, 26.96, 16.12	33.04, 34.29, 16.17	
1 1 2	0.1929	-0.395	41,37,30.00	-0.045, 0.014,-0.999	36.70, 24.36, 19.37	45.70, 27.59, 16.44	32.50, 32.59, 15.56	
1 1 3	0.2560	2.072	35,17,22.00	-0.336, 0.426,-0.840	33.36, 33.04, 22.36	27.05, 26.35, 18.44	39.43, 31.94, 14.49	
1 1 4	0.2623	-0.397	39,36,30.00	-0.016, 0.022,-1.000	37.07, 24.47, 19.32	46.14, 27.74, 16.68	32.96, 32.73, 15.47	
1 1 5	0.2632	3.492	51,46,26.00	-0.532,-0.367,-0.763	32.66, 21.94, 21.10	23.42, 25.52, 22.45	29.26, 12.64, 22.55	
2 2 0	0.0804	-0.742	38,46,26.00	0.328,-0.409,-0.852	42.85, 26.32, 23.48	50.78, 31.41, 20.12	36.77, 33.35, 19.79	
2 2 1	0.1447	1.810	41,42,28.00	0.120,-0.246,-0.962	37.67, 36.87, 18.64	36.02, 27.09, 17.42	46.85, 34.90, 22.09	
2 2 2	0.1463	-0.599	37,45,26.00	0.242,-0.428,-0.871	41.17, 26.11, 23.14	49.66, 30.09, 19.65	36.08, 34.04, 19.81	
2 2 3	0.1502	1.693	51,42,26.00	-0.367,-0.175,-0.914	31.16, 37.62, 21.19	29.44, 27.84, 20.04	40.89, 36.12, 19.45	
2 2 4	0.1601	-0.256	35,44,26.00	0.140,-0.427,-0.893	38.39, 26.52, 22.48	47.88, 27.77, 19.57	36.28, 35.86, 19.61	
2 2 5	0.1608	1.774	41,30,26.00	-0.175, 0.577,-0.798	34.91, 46.77, 25.05	30.47, 41.39, 17.89	43.76, 42.90, 22.47	
3 3 0	0.1716	4.613	45,28,23.00	-0.365, 0.710,-0.602	47.18, 31.09, 15.46	48.38, 36.35, 23.88	37.44, 30.07, 17.49	
3 3 1	0.1782	-0.298	53,43,25.00	-0.282,-0.360,-0.889	39.45, 20.36, 23.34	48.76, 23.31, 21.18	35.84, 28.75, 19.27	
3 3 2	0.1829	1.515	25,29,26.00	0.215, 0.221,-0.951	55.08, 41.52, 20.60	55.62, 31.54, 20.27	65.05, 42.03, 21.28	
3 3 3	0.1861	4.663	26,28,26.00	0.119, 0.278,-0.953	55.56, 21.76, 15.52	54.55, 30.71, 19.86	45.96, 22.04, 12.71	
3 3 4	0.1947	4.382	42,42,28.00	0.057,-0.194,-0.979	57.34, 15.42, 20.17	53.96, 24.83, 19.92	48.04, 12.05, 18.65	
3 3 5	0.2014	1.774	32,30,27.00	-0.012, 0.418,-0.909	54.86, 43.46, 23.37	53.01, 33.93, 20.96	64.47, 42.22, 20.90	
4 4 0	0.1571	1.798	42,20,13.00	-0.603, 0.685,-0.408	20.47, 15.70, -3.69	12.92, 12.40, -9.35	24.00, 20.92,-11.45	
4 4 1	0.1677	3.719	30,23,23.00	-0.111, 0.343,-0.933	6.44, 17.92, 7.96	-0.62, 24.83, 6.40	2.01, 11.89, 1.33	
4 4 2	0.1806	0.679	34,39,27.00	0.100,-0.251,-0.963	43.58, 15.19, 8.15	50.00, 7.86, 5.94	48.69, 21.45, 2.26	
4 4 3	0.1807	4.117	27,25,24.00	-0.060, 0.333,-0.941	7.46, 24.57, 9.29	3.34, 33.56, 7.84	0.54, 20.44, 3.37	
4 4 4	0.1812	2.070	36,46,24.00	0.177,-0.476,-0.861	29.52, -2.73, 16.37	24.83,-11.44, 14.87	38.35, -7.32, 15.36	
4 4 5	0.1814	3.719	29,23,23.00	-0.108, 0.341,-0.934	6.46, 17.91, 7.91	-0.39, 24.82, 6.31	2.05, 11.87, 1.27	
5 5 0	0.1008	3.266	34,33,28.00	0.019, 0.222,-0.975	12.83, 35.62, 12.53	4.19, 38.79, 7.53	12.12, 27.09, 7.31	
5 5 1	0.1124	-0.328	25,30,24.00	0.256, 0.195,-0.947	37.53, 32.78, 15.53	46.81, 35.36, 12.85	34.30, 41.94, 13.18	
5 5 2	0.1168	-0.325	27,32,27.00	0.241, 0.175,-0.955	37.38, 32.52, 15.21	46.63, 35.01, 12.32	34.15, 41.65, 12.73	
5 5 3	0.1174	-0.578	40,25,23.00	-0.336, 0.563,-0.755	29.04, 41.24, 13.65	34.66, 48.17, 9.14	23.19, 48.42, 17.40	
5 5 4	0.1179	3.542	49,35,26.00	-0.401, 0.265,-0.877	9.23, 38.33, 18.81	0.61, 43.35, 18.08	4.90, 30.30, 14.72	
5 5 5	0.1189	-0.340	27,31,27.00	0.199, 0.221,-0.955	36.89, 33.29, 14.82	45.96, 36.12, 11.69	33.36, 42.44, 12.87	

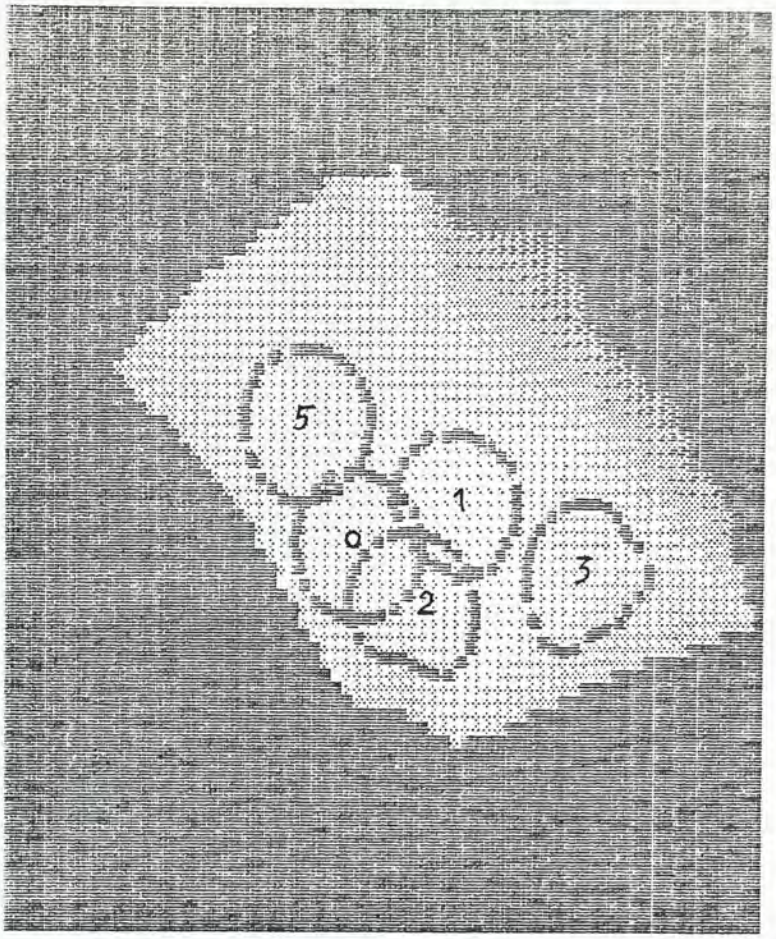
Fig 3.4.6 (b) The match table.

n	expected centre on scene		
0	31.6932	39.8676	25.7594
1	40.7646	36.3811	29.5741
2	37.5152	45.1661	25.9321
3	53.4786	43.1348	24.9112
4	30.9939	17.7455	21.6212
5	25.2003	30.9444	26.2282

n	expected orientation on scene		
0	0.0719	-0.2836	-0.9557
1	-0.0355	0.0699	-0.9969
2	0.2754	-0.4154	-0.8669
3	-0.3342	-0.3271	-0.8839
4	0.1407	0.2306	-0.9628
5	0.2713	0.1298	-0.9537

	expected reference on scene		
0	39.9826	35.4524	16.9564
1	48.6122	30.8295	14.1990
2	35.4301	43.8998	13.4803

(c)



(d)

Sub Template :	0	1	2	3	4	5
Match no :	0	2	2	1	-1	5

	position	orientation		reference										
0 0	32,40,26.00	0.069,-0.332,-0.941	*	39.90	29.29	20.25	*	48.84	32.70	17.35	*	35.56	37.48	16.50
1 2	41,37,30.00	-0.045, 0.014,-0.999	*	36.70	24.36	19.37	*	45.70	27.59	16.44	*	32.50	32.59	15.56
2 2	37,45,26.00	0.242,-0.428,-0.871	*	41.17	26.11	23.14	*	49.66	30.09	19.35	*	36.06	34.04	17.81
3 1	53,43,25.00	-0.282,-0.360,-0.889	*	39.45	20.36	23.34	*	48.76	23.31	21.18	*	35.24	28.75	19.27
5 5	27,31,27.00	0.199, 0.221,-0.955	*	36.89	33.27	14.82	*	45.96	36.12	11.69	*	33.36	42.44	12.87

mean ref. 38.82 26.68 20.18 47.79 29.96 17.26 34.67 35.96 16.80

standard deviation : 9.7945

Fig 3.4.6 cont'd.. Result of recognition using the same set of subtemplates as shown in fig.3.4.5a.
(c) Expected results;
(d) Matched subtemplates;
Note that subtemplate 4 was lost.

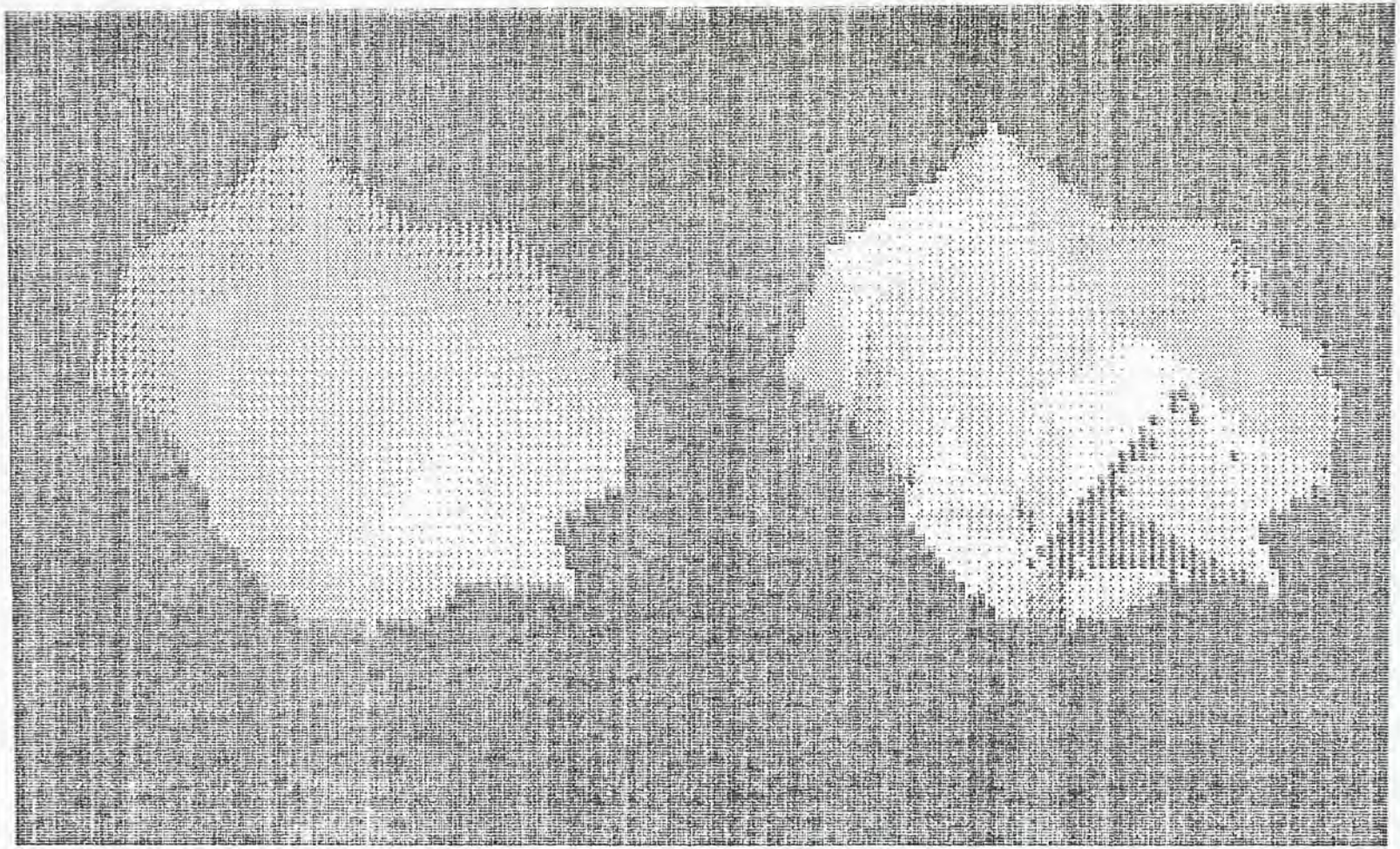


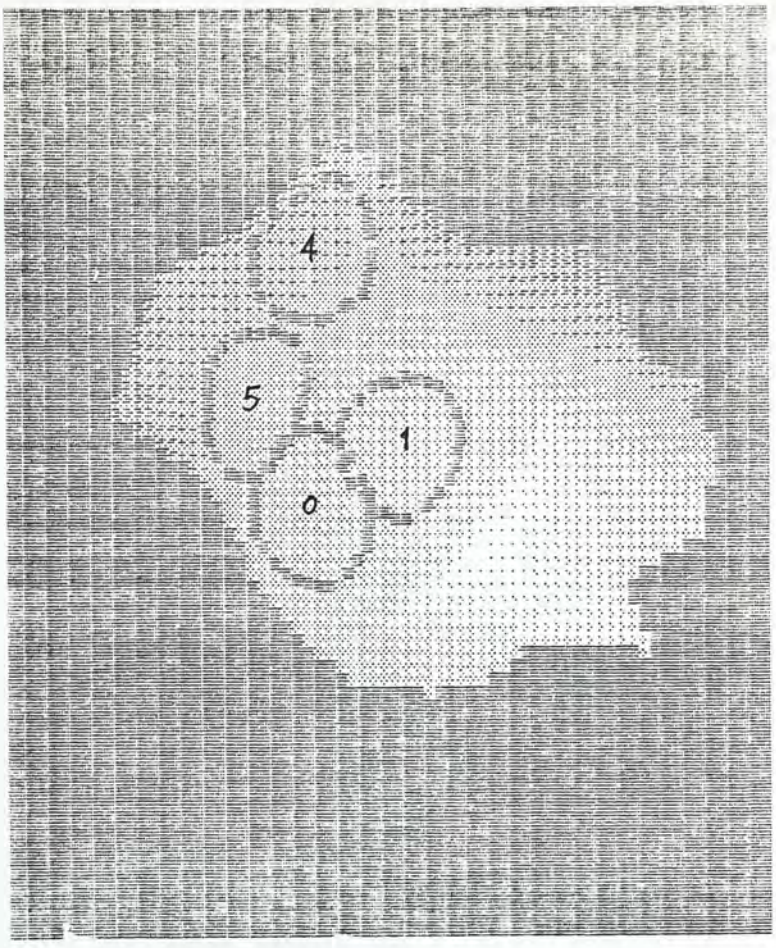
Fig 3.4.7 (a) Stone partially occluded by another object.
Left : depth image; right : intensity image.

n	expected centre on scene		
0	27.5947	37.0710	27.6537
1	34.1026	31.5045	33.6981
2	33.0658	41.8193	31.3052
3	47.3172	37.2564	36.1736
4	27.5927	16.6209	18.2197
5	21.0555	28.2798	25.3053

n	expected orientation on scene		
0	0.4451	-0.0885	-0.8911
1	0.3314	0.2795	-0.8812
2	0.5876	-0.3620	-0.7656
3	0.0440	-0.9848	-0.9854
4	0.5854	0.4024	-0.7426
5	0.9457	0.2841	-0.7088

	expected reference on scene		
0	38.2180	30.5040	21.7220
1	47.5918	36.0155	23.7072
2	36.2329	42.9778	19.2106

(b)



(c)

Sub-Template : 0 1 2 3 4 5
Match no : 2 4 -1 -1 0 0

	position	orientation		reference	ϕ			1			2		
0 2	28.37,28.00	0.449,-0.089,-0.889	*	42.94	31.21	22.30	*	52.09	34.85	24.92	*	39.93	40.22	20.09
1 4	35.33,24.00	0.357, 0.288,-0.888	*	41.78	29.04	19.92	*	51.32	30.48	21.28	*	39.77	37.55	17.09
4 0	22.18,19.00	0.423, 0.413,-0.807	*	35.33	33.58	19.20	*	46.11	35.87	18.93	*	34.27	43.13	16.24
5 0	22.29,24.00	0.623, 0.332,-0.709	*	40.25	32.62	22.33	*	49.92	34.81	24.65	*	38.50	42.40	21.55
mean ref.	40.34	31.37	21.09	49.85	33.83	22.35	35.14	40.83	18.90					
standard deviation : 6.6667														

Fig 3.4.7 cont'd.. Final result when stone partially occluded.
Compared with fig.3.4.5f, two subtemplates are missing.

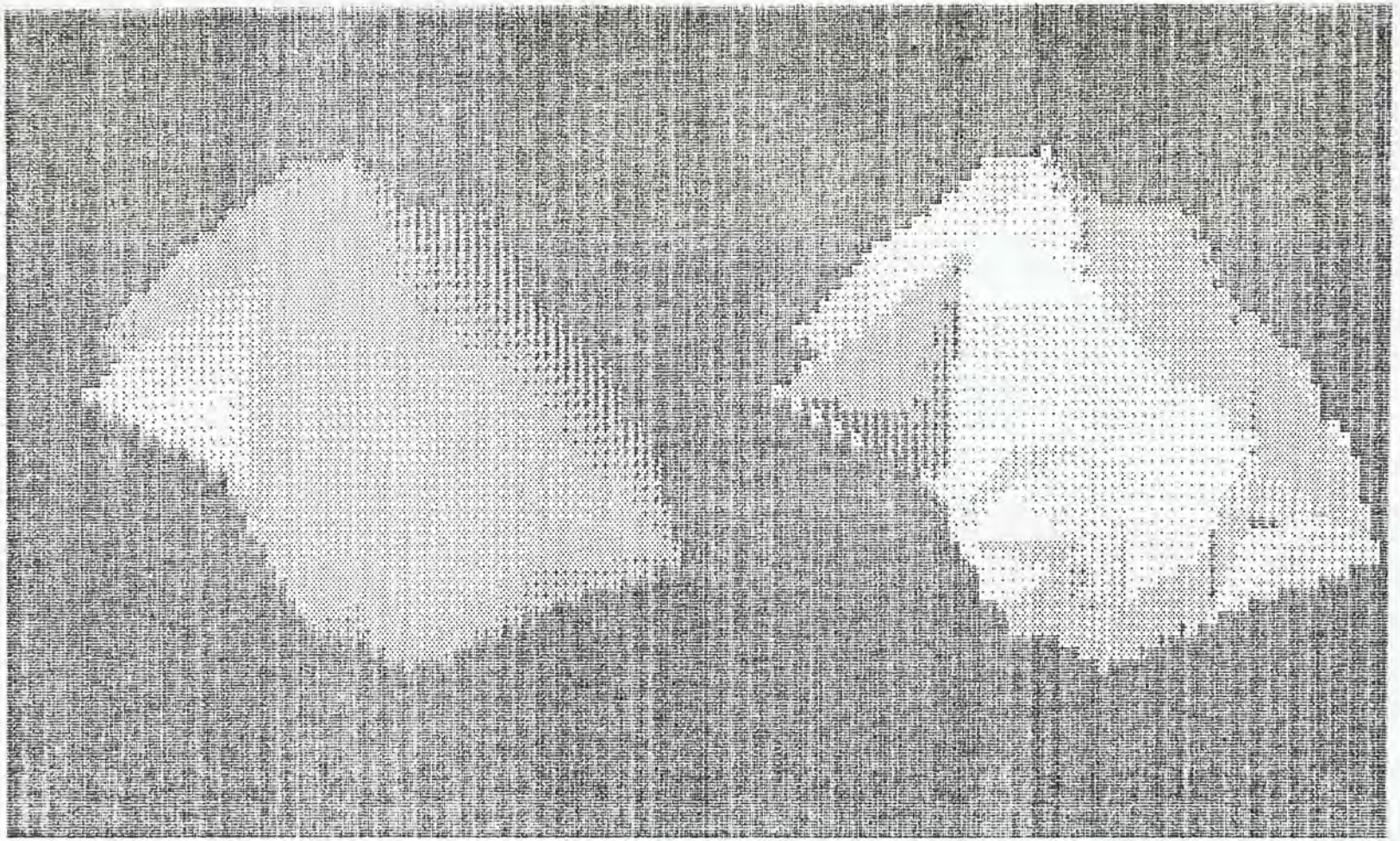


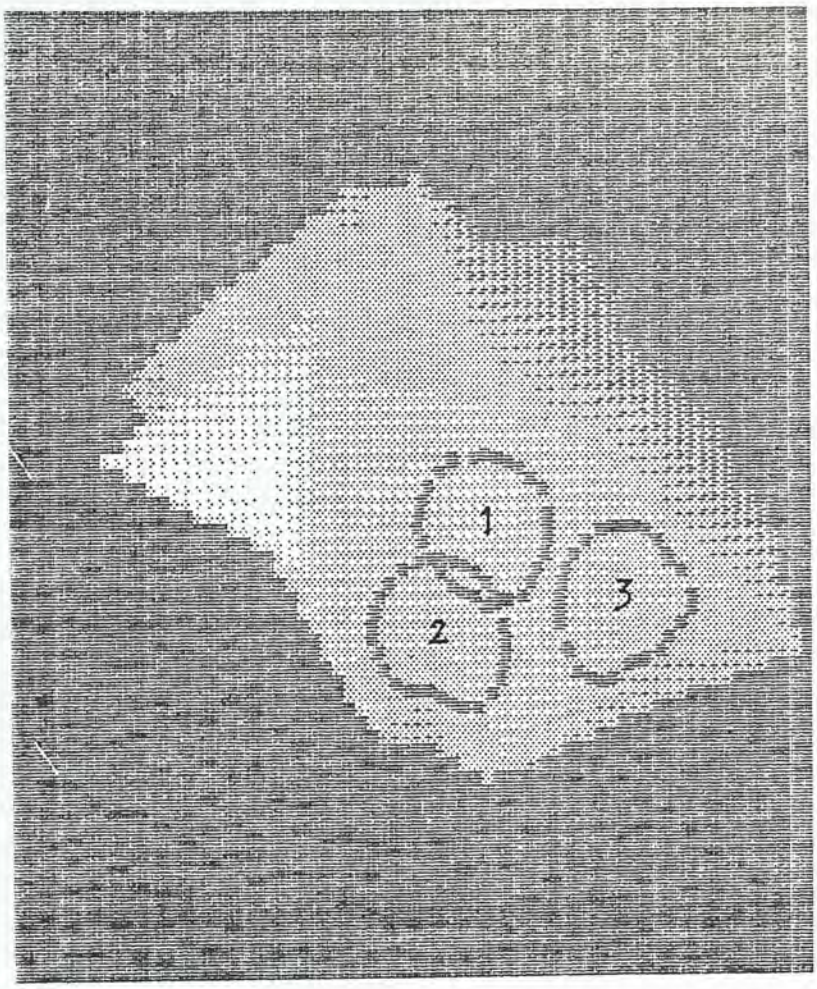
Fig 3.4.8 (a) Partial occlusion - another example.

n	expected centre on scene		
0	31.6982	39.8676	25.7594
1	40.7545	36.3811	29.5741
2	37.5152	45.1661	25.9321
3	53.4736	42.1348	24.9112
4	30.9939	17.7455	21.6212
5	25.2003	30.9444	26.2282

n	expected orientation on scene		
0	0.0719	-0.2956	-0.9337
1	-0.0255	0.0699	-0.9989
2	0.2754	-0.4154	-0.8669
3	-0.3342	-0.3271	-0.8832
4	0.1407	0.2206	-0.9628
5	0.2713	0.1298	-0.9537

	expected reference on scene		
0	39.5835	35.4524	16.9564
1	46.6122	30.8285	14.1980
2	35.4601	43.8998	13.4807

(b)



(c)

SubTemplate :	0	1	2	3	4	5
Match no :	-1	2	2	1	-1	-1

	position	orientation		reference	p			1.			2		
1 2	41,37,30.00	-0.065, 0.014,-0.997	*	36.70	24.36	19.37	*	45.70	27.59	16.44	*	32.50	32.59	15.56
2 2	37.45,26.00	0.242,-0.428,-0.871	*	41.17	26.11	23.14	*	49.66	30.09	19.65	*	36.08	34.04	19.81
3 1	53,43,25.00	-0.282,-0.360,-0.889	*	39.45	20.36	23.34	*	48.76	23.31	21.18	*	33.84	28.75	19.27

mean ref. 39.11 23.61 21.95 48.04 27.00 19.09 34.81 31.80 18.21

standard deviation : 5.1931

Fig 3.4.8 cont'd.. Results. Three out of six subtemplates are labeled sucessfully.

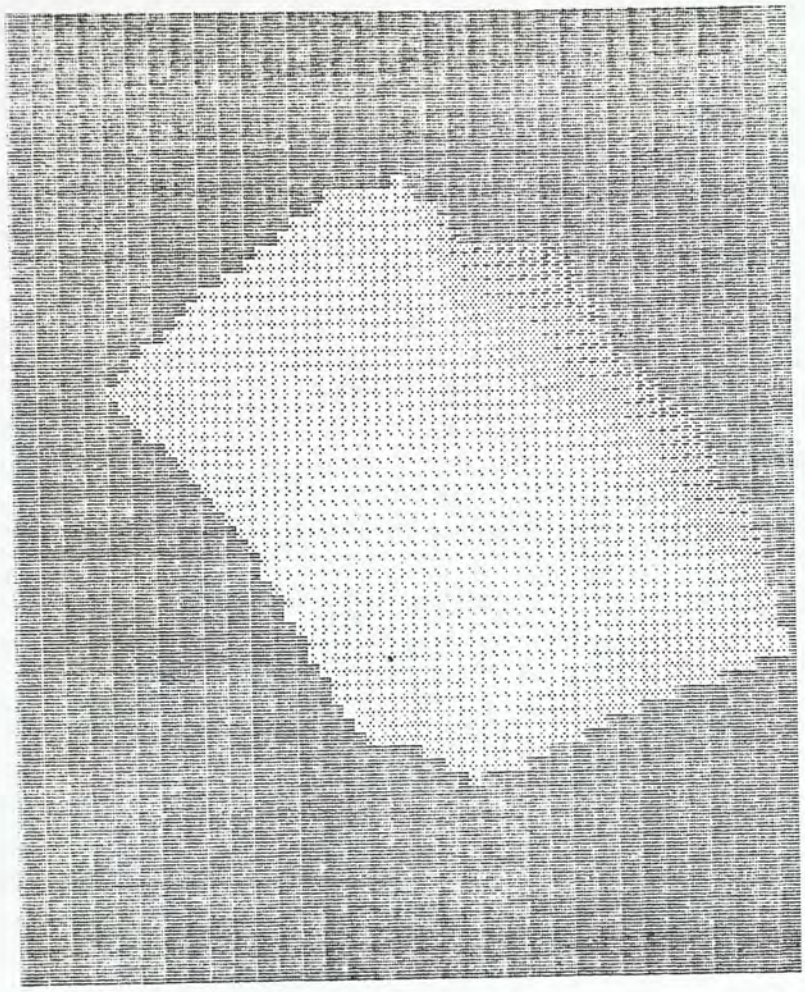
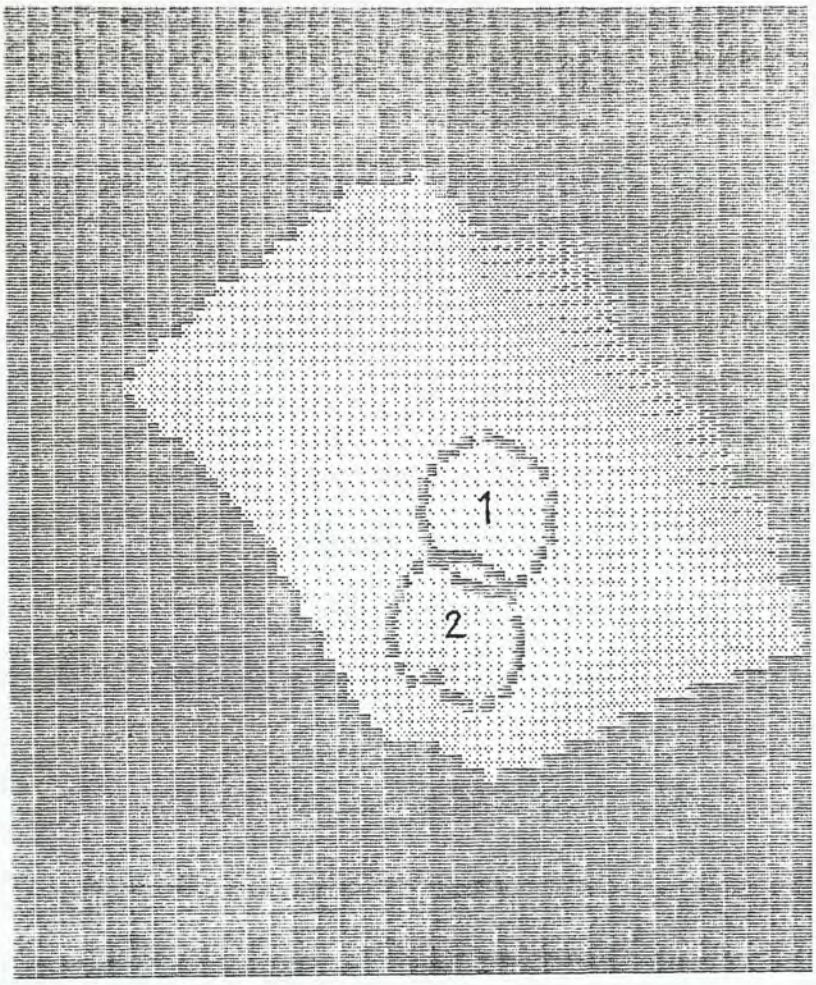


Fig 3.4.9 (a) The scene shown in fig.3.4.6a corrupted by 6% random noise.

	error	rotation	position	orientation	reference ϕ			1			2		
0 0 0	0.2131	-0.142	53,17,-3.00	-0.312, 0.280,-0.908	34.88,	33.71,	16.60	43.21,	36.05,	11.94	33.57,	43.26,	19.36
0 0 1	0.2265	2.393	50,38,26.00	-0.284, 0.112,-0.952	22.03,	34.38,	21.16	14.40,	27.97,	20.34	28.32,	27.32,	17.91
0 0 2	0.2557	1.612	26,25,24.00	-0.043, 0.314,-0.949	26.78,	31.87,	17.63	26.15,	23.32,	12.49	36.73,	31.76,	16.60
0 0 3	0.2629	3.170	32,19,22.00	0.055, 0.166,-0.985	26.87,	40.35,	19.28	17.27,	41.10,	16.61	26.72,	30.59,	17.11
0 0 4	0.2718	0.122	51,16,-3.00	-0.259, 0.354,-0.899	34.12,	33.08,	16.52	42.81,	33.17,	11.53	35.58,	42.53,	19.28
0 0 5	0.2822	3.197	31,19,22.00	-0.027, 0.209,-0.977	26.17,	40.86,	20.06	16.39,	41.68,	16.27	25.63,	31.21,	17.48
0 0 6	0.2964	2.547	50,40,26.00	-0.270,-0.013,-0.963	22.14,	34.35,	21.02	13.74,	28.92,	21.14	27.45,	26.09,	19.12
0 0 7	0.2970	0.752	35,42,26.00	0.097,-0.436,-0.895	34.00,	22.98,	23.03	41.11,	16.14,	24.65	40.96,	29.50,	20.03
0 0 8	0.2989	2.237	30,23,23.00	-0.107, 0.351,-0.930	23.77,	35.81,	19.51	17.35,	29.33,	15.41	31.44,	30.41,	16.05
1 1 0	0.2909	2.886	39,34,30.00	-0.086, 0.210,-0.974	37.91,	30.32,	19.95	28.23,	28.92,	17.86	40.31,	22.74,	13.89
1 1 1	0.3026	-0.221	40,35,30.00	-0.096, 0.130,-0.987	35.74,	25.98,	19.39	44.99,	27.93,	16.15	32.87,	35.18,	16.73
1 1 2	0.3153	4.794	36,18,23.00	-0.412, 0.304,-0.859	33.85,	28.53,	20.15	34.66,	38.49,	20.30	23.89,	29.34,	20.44
1 1 3	0.3196	2.930	36,35,30.00	0.053, 0.055,-0.997	39.72,	28.34,	19.53	30.32,	27.02,	16.38	42.41,	19.79,	15.09
1 1 4	0.3213	2.918	51,34,27.00	-0.377, 0.364,-0.852	34.31,	32.23,	21.44	24.42,	30.98,	22.26	34.48,	25.78,	13.79
1 1 5	0.3258	-0.228	41,36,30.00	-0.064, 0.089,-0.994	36.17,	25.47,	19.34	45.52,	27.35,	16.33	33.34,	34.53,	16.20
1 1 6	0.3291	-0.230	41,34,30.00	-0.143, 0.239,-0.960	35.16,	27.38,	19.59	44.24,	29.77,	16.16	32.13,	36.80,	18.11
1 1 7	0.3390	3.490	51,46,27.00	-0.520,-0.344,-0.782	32.84,	22.26,	20.89	23.59,	25.84,	22.18	29.39,	12.94,	22.03
1 1 8	0.3418	3.388	50,46,27.00	-0.505,-0.330,-0.798	32.90,	22.71,	20.72	23.35,	25.02,	22.59	30.61,	12.98,	21.05
2 2 0	0.1743	-0.581	39,33,28.00	-0.098, 0.381,-0.920	37.01,	36.11,	18.69	44.44,	42.56,	16.89	32.22,	43.12,	23.98
2 2 1	0.1815	3.510	41,27,24.00	-0.302, 0.589,-0.750	39.70,	45.69,	22.70	30.57,	49.27,	24.66	35.67,	37.01,	19.80
2 2 2	0.1877	0.812	35,32,28.00	-0.022, 0.359,-0.933	33.69,	40.25,	20.39	40.27,	35.32,	14.71	41.10,	45.78,	24.19
2 2 3	0.1913	-0.563	38,45,27.00	0.291,-0.389,-0.874	41.60,	26.52,	22.82	50.34,	30.52,	20.07	36.84,	34.69,	19.57
2 2 4	0.2039	0.377	27,27,26.00	0.031, 0.352,-0.935	34.93,	38.25,	19.38	43.91,	36.42,	15.36	38.73,	46.08,	24.30
2 2 5	0.2135	3.121	38,25,24.00	-0.275, 0.497,-0.823	38.96,	45.79,	22.77	28.99,	46.34,	22.32	38.60,	36.41,	19.32
2 2 6	0.2249	-1.200	48,29,22.00	-0.414, 0.771,-0.484	34.95,	43.44,	21.55	38.22,	52.05,	25.45	28.48,	42.47,	29.11
2 2 7	0.2262	0.958	34,33,28.00	0.054, 0.235,-0.971	34.67,	39.47,	19.74	40.55,	32.92,	15.00	42.75,	44.27,	23.14
2 2 8	0.2277	3.289	40,24,22.00	-0.424, 0.550,-0.719	37.59,	46.09,	23.21	27.93,	47.57,	25.32	35.44,	36.91,	19.85
3 3 0	0.1591	1.537	24,28,26.00	0.194, 0.175,-0.965	55.09,	40.97,	19.98	55.39,	30.97,	20.06	65.08,	41.26,	20.38
3 3 1	0.1953	0.202	54,29,15.00	-0.571, 0.738,-0.359	37.52,	36.49,	34.32	45.32,	37.09,	28.08	41.21,	44.11,	39.65
3 3 2	0.1982	1.349	43,27,23.00	-0.381, 0.677,-0.630	46.93,	44.70,	30.95	48.19,	36.99,	24.72	55.43,	48.78,	27.63
3 3 3	0.1988	-0.061	27,18,22.00	0.185, 0.569,-0.801	45.29,	36.10,	18.15	54.54,	35.12,	21.81	44.56,	45.13,	22.39
3 3 4	0.2034	4.688	25,27,26.00	0.082, 0.250,-0.965	54.68,	21.27,	15.60	54.05,	30.35,	19.74	44.98,	21.68,	13.22
3 3 5	0.2077	1.564	25,28,26.00	0.167, 0.211,-0.963	54.97,	41.42,	20.46	55.06,	31.42,	20.12	64.97,	41.50,	20.54
3 3 6	0.2095	1.430	39,19,19.00	-0.722, 0.447,-0.527	43.83,	43.17,	30.37	44.06,	34.26,	25.85	49.37,	47.05,	23.00
3 3 7	0.2139	3.450	52,30,20.00	-0.507, 0.778,-0.371	53.14,	40.30,	19.13	47.21,	37.81,	26.78	47.09,	35.42,	12.84
3 3 8	0.2241	4.562	42,40,29.00	-0.046,-0.165,-0.985	54.12,	15.29,	19.59	52.55,	25.16,	19.80	44.28,	13.74,	18.67
4 4 0	0.1836	0.977	49,26,14.00	-0.553, 0.714,-0.429	31.60,	25.71,	-1.05	29.92,	24.96,-10.38		35.87,	34.64,	-2.46
4 4 1	0.2022	-1.169	50,37,26.00	-0.321, 0.199,-0.926	17.38,	35.31,	11.91	20.46,	43.84,	7.70	7.87,	38.22,	10.85
4 4 2	0.2098	1.815	41,21,17.00	-0.687, 0.612,-0.392	18.63,	13.85,	-2.95	10.80,	9.65,	-7.54	21.00,	18.64,	-11.41
4 4 3	0.2118	2.188	54,46,22.00	-0.338,-0.649,-0.682	25.41,	-2.81,	20.17	19.04,	-9.86,	23.28	31.72,	-9.90,	17.03
4 4 4	0.2304	2.691	24,24,24.00	-0.026, 0.239,-0.971	17.57,	3.64,	2.73	9.00,	2.05,	-2.18	22.34,	-2.41,	-3.65
4 4 5	0.2403	3.118	40,46,26.00	0.398,-0.328,-0.857	16.04,	1.75,	6.14	9.88,	0.57,	-1.64	18.55,	-7.91,	5.62
4 4 6	0.2412	4.333	28,27,25.00	-0.001, 0.379,-0.925	9.37,	28.39,	10.00	7.24,	38.11,	9.00	1.65,	26.09,	4.08
4 4 7	0.2466	4.368	47,27,18.00	-0.517, 0.738,-0.433	9.77,	25.67,	26.14	8.21,	33.97,	31.50	-0.11,	24.44,	25.17
4 4 8	0.2516	-0.526	52,42,25.00	-0.303,-0.250,-0.919	26.94,	30.42,	0.96	33.23,	32.20,	-6.81	20.07,	36.24,	-3.38
5 5 0	0.1557	-0.589	41,26,24.00	-0.337, 0.604,-0.722	29.03,	41.85,	14.34	34.64,	49.03,	10.22	23.17,	48.81,	18.48
5 5 1	0.1575	2.545	44,26,22.00	-0.415, 0.690,-0.593	7.07,	33.19,	18.39	-2.52,	32.87,	15.59	9.89,	31.40,	8.96
5 5 2	0.1593	3.734	47,34,27.00	-0.342, 0.365,-0.866	11.33,	41.03,	19.17	3.86,	47.67,	18.35	5.50,	34.13,	14.88
5 5 3	0.1653	-0.565	42,28,25.00	-0.278, 0.625,-0.730	30.04,	41.68,	14.78	36.13,	48.66,	10.99	24.29,	48.85,	18.73
5 5 4	0.1673	-1.112	35,20,23.00	-0.306, 0.209,-0.929	25.95,	41.01,	12.27	28.87,	49.19,	7.31	16.63,	44.61,	12.72
5 5 5	0.1754	-0.860	40,27,25.00	-0.264, 0.570,-0.778	28.87,	43.25,	16.19	33.86,	51.51,	13.58	21.01,	48.83,	18.35
5 5 6	0.1777	3.199	47,31,25.00	-0.361, 0.609,-0.706	8.17,	38.05,	21.39	-1.25,	41.42,	21.43	8.19,	32.84,	18.21
5 5 7	0.1806	3.350	49,34,26.00	-0.382, 0.397,-0.835	8.44,	37.66,	19.70	-0.74,	41.60,	19.29	5.52,	30.23,	13.68
5 5 8	0.1838	2.899	40,44,29.00	0.236,-0.279,-0.931	14.89,	24.29,	9.93	8.05,	21.41,	3.29	18.54,	14.98,	10.22

Fig 3.4.9 (b) The resulting match table. Compare to that in fig.3.4.6b.



Sub-Template : 0 1 2 3 4 5

Match no : -1 5 3 -1 -1 -1

	position	orientation		reference	ϕ		1.			2		
1 5	41,36,30.00	-0.064, 0.089,-0.994	*	36.17	25.47	19.34	*	45.52	27.35	16.38	*	33.84	34.53	16.20
2 3	39,45,27.00	0.291,-0.389,-0.874	*	41.60	26.52	22.82	*	50.34	30.52	20.07	*	35.84	34.69	19.57
mean ref.	38.88	25.99	21.08	47.93	28.93	18.20		35.09	34.61	17.39				

standard deviation : 5.3273

Fig 3.4.9 (c) Matching result with the presence of 6% random noise.

so that it can accomodate matches with larger matching error, and let the DP scheme to select the desired ones and reject the false matches. However as the number of entries for each subtemplate increases, there are occasions when the false matches themselves form a compatible labeling set which results in an incorrect interpretation.

For this and other reasons, it deserves to direct future research on making the DP scheme more sophisticated. One way to do that is to re-formulate the global compatibility as

$$c(.) = c_1(x_1,x_2,x_3) + c_2(x_2,x_3,x_4) + c_3(x_3,x_4,x_5) + \ldots . \quad (3.19)$$

instead of the former hypothesis we adopted, ie.

$$c(.) = c_1(x_1,x_2) + c_2(x_2,x_3) + c_3(x_3,x_4) + \ldots . \quad (3.20)$$

That is instead of considering the overall compatibility in terms of pairwise compatibility of individual subtemplates, we can work on triples of subtemplates. One may generalize that to quardruple and so on, but the computational effort increases exponentially, and we believe that 3 is an optimal choice regarding performance versus complexity.

In addition to the enhancement of the DP scheme, a more sophisticated subtemplate matching stage that retains as little probable matches as possible can do much help in improving the overall performance of the scheme. One obvious way to achieve this is to go one step further in matching the internal points of the subtemplate after matching the outer boundary, provided that the result in matching the boundary alone is good enough. There-

fore one adds a final step in the matching hierarchy to match the internal points based on previous matching results. Note that we do not have to rotate the subtemplate any more, since the alignment is already complete after matching the outer boundary.

So far in the 3-D case we have not yet considered the application of a weighting scheme that reflects the relative importance of the subtemplates as we did in the 2-D case. That does not mean that the weighting scheme is not applicable, but rather such consideration is even more important in the 3-D case. Currently the selection of subtemplates is done manually. To achieve a fully automatic recognition system we would like it to select by itself those subtemplates that are considered as important. However the implementation may turn out to be a bit more difficult than the 2D case, and we leave it for further research.

The current algorithm is implemented on an IBM PC/AT and is coded in Pascal. To recognize the stone using 6 subtemplates as shown in the experiments takes about 45 minutes, with most of the computing time spent in subtemplate matching which spans a search region of about 30 by 30 which is a total of 900 points. To process one single point on the scene takes on the average something like 3 seconds. The most time consuming part in that is the extraction of the space curve, and we believe that with the proper hardware built this computing time can be greatly reduced. Currently the Fourier Transform is an un-optimized recursive version written in Pascal and not much attention has been paid in global optimization of the implementation.

One obvious way to reduce the recognition time is to restrict the set of points on the scene that is being matched. For example one may select subtemplates that are centered at points of high curvature. When actual matching is done the scene is preprocessed to give points of high curvature, and matching is done on that restricted set of points.

香港中文大學圖書館藏書

CHAPTER 4

CONCLUSION

A method for recognizing partially-occluded, arbitrarily-shaped objects for both two and three dimensions has been presented. There are two principal components to the method. The first is a subtemplate matching scheme that breaks the object template down into parts for matching. For the 2-D case the well-known matching-in-θ-s-space method is used in matching the subtemplates with boundaries of the object from the scene. For the 3-D case a new algorithm that works on depth map representation of the object is proposed. The algorithm attempts to estimate the orientation of a surface patch and performs the matching using mainly the outer boundary of the patch only. The second main component of the method is a constraint application process that selects among a number of probable matches that result from the subtemplate matching stage a most consistent set of labels to form the solution. Such a constraint application process is formulated as a dynamic programming problem in this work.

Preliminary experiment shows that the method works well on objects that are truly arbitrarily-shaped. As already mentioned in section 3.1, the scheme here is supposed to work in co-operation with other techniques that are specially designed for the matching of regular shapes such as planes, cylinders and cones to form a model-driven recognition system, instead of a data-driven system that try to extract from the scene data some

global and common entites(planes,cylinders etc). The reason for this is that, as we believe, many real-world objects that we are interested in consist of some parts that are truly arbitrarily-shaped, and cannot be characterised by any simple geometric entities. However those irregular parts do to a certain extent help us to identify an object. Therefore we prefer a model-driven scheme which makes use of highly constraining object-specific features.

Compared with previous methods of object recognition in 2-D the required time to obtain a solution is much shorter using our algorithm. The same is true for the 3-D case. Less restrictions such as the presence of planar surfaces are put on the shape of the object. Also the size of our library is relatively small compared to schemes that use view-dependent representation of the objects.

For our 2-D scheme scaling is not allowed. As a matter of fact most schemes that work directly on the pixel level do not allow scaling since variation in size usually means an additional dimension for the search space. However for most industrial applications the distance between the object to be recognized and the camera is usually known. For the 3-D case since the input comes from a range-finder, the scaling problem does not exist.

Regarding susceptability to noise, the algorithm may not be as robust as other schemes that based on segmenting the object into high level entities. However one must bear in mind that in a

certain way the robustness of a scheme is a trade-off between sacrificing the ability of the scheme to handle real fine details and its susceptability to noise. Also since the data obtained using range-finders are fairly accurate, the algorithm works well in the usual case. The ways of improvement proposed in section 3.4 to make the subtemplate matching as well as the DP scheme more sophisticated can to a certain extent improve the overall quality of the scheme as well as its robustness.

There are still things remained to be done. Quantitative methods of choosing the various constants such as the error threshold in the subtemplate matching scheme and others in the DP stage are yet to be determined. The recognition time can be reduced if we restrict the points on the scene for matching to a smaller subset. For example we may define the subtemplates to be centered on points of maximum curvature and corresponding search through only those sharp points in the incoming scene. With the proper hardware built and preferably parallel implementation in the subtemplate matching process, the method may be useful for real-time applications.

REFERENCES

[1] H. Blum, "A transformation for extracting new descriptors of shape," in Models for the perception of Speech and Visual Form, W. Wathen-Dunn, Ed. Cambridge, MA: M.I.T. Press, 1967, PP.362-380.

[2] R.O. Duda and P.E. Hart, "Use of the Hough Transform to detect lines and curves in pictures," Communs Ass. Comput. Mach. 15, 11-15(1975).

[3] E. Persoon and K.S. Fu, "Shape Discrimination Using Fourier Descriptors," IEEE Trans. Systems, Man & Cybernetics, vol.SMC-7, No.3, March 1977.

[4] S.A. Dudani, K.J. Breeding and R.B. McGhee, "Aircraft Identification by Moment Invariants," IEEE Trans. on Compt., vol. c-26, No.1, January 1977.

[5] J. Sklansky, "On the Hough Technique for curve detection," IEEE Trans Comput., vol.C-27, PP.126-143, Feb.1978.

[6] W.S. Rutkowski, S.Peleg and A.Rozenfeld, "Shape Segmentation Using Relaxation," IEEE Trans on Patt. Anal. & Mach. Int., vol. PAMI-3, No.4, July 1981.

[7] J.W.McKee and J.K.Aggarwal, "Computer Recognition of Partial Views of Curved Objects," IEEE Trans. on Comput., vol. c-26, No. 8, August 1977.

[8] W.A. Perkins, "A Model-Based Vision System for Industrial Parts," IEEE Trans. on Comput., vol. c-27, No. 2, Feb 1978.

[9] Bir Bhanu and Oliver D. Faugeras, "Shape Matching of Two-Dimensional Objects," IEEE Trans on Patt. Anal. & Mach. Int., vol. PAMI-6, No.2, March 1984.

[10] D.H.Ballard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," Pattern Recogn., vol.13, no.2, pp.111-122, 1981.

[11] J.L.Turney, T.N. Mudge and R.A. Volz, "Recognizing Partially Occluded Parts," IEEE Trans. on Pat. Anal. & Mach. Int., vol- PAMI-7, No.4, July 1985.

[12] D.H. Ballard and J. Sklansky, "A Ladder-Structured Decision Tree for Recognizing Tumors in Chest Radiographs," IEEE Trans. on Comput., vol.c-25, No.5, May 1976.

[13] T. Pavlidis, "Algorithms for Graphics and Image Processing," pp.142-144, Computer Science Press, Inc, 1982.

[14] H.G. Barrow and R.J. Popplestone, "Relational Descriptions in Picture Processing," Machine Intell., vol.6, pp.377-396, 1971.

[15] R. Bellman and S. Dreyfus, "Applied Dynamic Programming," Princeton, NJ: Princeton University Press, 1962.

[16] C.H. Lee & A. Rosenfeld, "An approximation technique for photometric stereo," Pattern Recognition Letters 2(1984) pp.339-343.

[17] K.Ikeuchi and B.K.P.Horn, "Numerical Shape from Shading and Occluding Boundaries," Artificial Intellegence...

[18] K. Ikeuchi, "Shape from Regular Patterns," Artificial Intellegence 22 (1984) pp.49-75.

[19] P.J.Besl & R.C.Jain, "Three-Dimensional Object Recognition," ACM Computing Surveys, Vol.17, No.1, March 1985.

[20] T.P.Wallace & P.A.Wintz, "An Efficient 3-D Aircraft Recognition Algorithm Using Normalized Fourier Descriptors," Computer Graphics & Image Processing, 13, pp.99-126 (1980).

[21] B.Cernuschi-Frias & D.B.Cooper, "3-D Space Location & Orientation Parameter Estimation of Lambertian Spheres & Cylinders From a Single 2-D Image By Fitting Lines & Ellipses to Thresholded Data," IEEE Trans. Pattern Anal. & Mach. Intell., Vol, PAMI-6, No.4, July 1984 pp.430-441.

[22] R.A.Brooks, "Model-based 3-D interpretations of 2-D images," IEEE Trans. Pattern Anal. Mach. Intell. PAMI-5, No.2, March 1983 pp.140-149.

[23] B.Bhanu, "Representation & Shape Matching of 3-D Objects," IEEE Trans. Pattern Anal. Mach. Intell., Vol. PAMI-6, No.3 May 1984 pp.340-351.

[24] D.H. Ballard & D. Sabbah, "Viewer Independent Shape Recognition," IEEE Trans. Pattern Anal. Mach. Intell., Vol. PAMI-5, No.6 November 1983 pp.653-659.

[25] O.D.Faugeras & M.Hebert, "The representation, Recognition & Positioning of 3-D Shapes From Range Data," pp.253-272, 4th Scandinavaian CConference on Image Analysis, 1985.

[26] M. Oshima & Y. Shirai, "Object Recognition Using 3-D Information," IEEE Trans. Pattern Anal. Mach. Intell., Vol. PAMI-5, No.4 July 1983 pp.353-361.

[27] B.K.P.Horn, "Extended Gaussian Images," Proceedings IEEE, Vol.72, No.12, Dec 1984 pp.1671-1686.

[28] M.J. Magee, B.A. Boyter, C.H. Chien & J.K. Aggarwal, "Experiments in Intensity Guided Range Sensing Recognition of 3-D Objects," IEEE Trans. Pattern Anal. Mach. Intell., Vol. PAMI-7, No.6 Nov. 1985 pp.629-637.

Appendix A Tracing the Space Curve

Given an object boundary surface in the form $z=f(x,y)$, the following algorithm traces a space curve ψ which is the intersection of the surface and a sphere centered at a surface point ζ_0 with radius r, ie. all the points on ψ are r away from ζ_0.

The algorithm is basically a conventional left-looking procedure in contour tracing. A slight modification on the tracer algorithm by T.Pavlidis[13] will suffice if the resolution being worked on is high. However if the resolution is low, severe error will result since the grid points on the surface are seldom exactly r from ζ_0. This usually causes some sort of zig-zag in the final outcome. To overcome this difficulty and to give a reasonable estimation even when part of the space curve is indeed invisible, interpolation between points is done to obtain the space curve.

The modified algorithm is as follows:

Notation: ψ is a list containing the space curve, A is the starting point of ψ, C is the current point whose neighbourhood is examined, S the search direction in terms of the code of Fig.A.1, *first* is a flag that is true only when the tracing starts, and *false* is a flag that is true when a next point on the space curve is found.

Procedure MODIFIED-TRACER

0. Choose a point A on the surface that is exactly r from ζ_B.
1. Set C to A, S to 6 and *first* to *true*.
2. While $C<>A$ or *first=true* do steps 3-12.
   Begin.
3. *found*:=*false*.
4. While *found* is *false* do steps 5-11, at most three times.
   Begin.
5. If B, the $(S-1)$-neighbor of C is within r from ζ_B then
   Begin.
6. Interpolate between B and the $(S-2)$-neighbour of C to obtain D.
   Enter D into ψ, C:=B, S:=$S-2$, *found*:=*false*.
   End.
7. Else If B, the S-neighbour of C is within r from ζ_B then
   Begin.
8. Interpolate between B and the $(S-1)$-neighbour of C to obtain D.
   Enter D into ψ, C:=B, *found*:=*false*.
   End.
9. Else If B, the $(S+1)$-neighbour of C is within r from ζ_B then
   Begin.
10. Interpolate between B and the S-neighbour of C to obtain D.
    Enter D into ψ, C:=B, *found*:=*false*.
    End of Algorithm.
11. Else S:=$S+2$.
    End.
12. *first*:=*false*.
    End.
13. End of Algorithm.

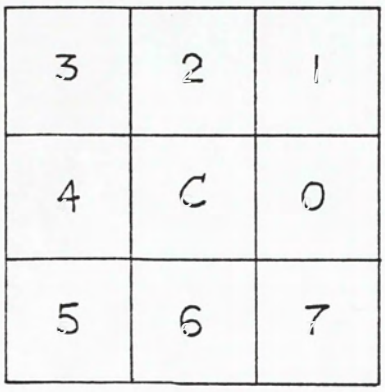


Fig A.1 Notation used for defining the relative locations of pixels with respect to a point C. (From Pavlidis[29] pp.134)


000471314